**海南大学计算机科学与技术学院**

**《计算机图形学》实验报告**

****

**班 级：计算机科学与技术2021-3班**

**成 员： 李季鸿 20213002624**

**指导老师： 高新瑞**

**完成日期： 2023年12月6日**

**实验得分：**

**《计算机图形学》创意实验**

**一、实验设计思路**

创建一个3D飞行汽车模型：使用Java 3D API构建一个具有动态特性的交互式3D模型。

增强用户交互：实现鼠标控制来操作模型的旋转、平移和缩放。

视觉效果优化：利用纹理映射、光照和材质来提升模型的视觉效果。

实验步骤

场景和视图设置：

创建一个Canvas3D并配置SimpleUniverse。

设置视图平台，以便于用户能够观察到整个3D场景。

模型构建：

使用BranchGroup和TransformGroup来组织和管理3D图形对象。

利用Bezier曲面（BezierThreeOrderSurfaceface）创建复杂的3D形状，为飞行汽车的各个部分建模。

纹理和材质应用：

加载并应用纹理来增强模型的视觉细节。

设置材质属性，如漫反射和高光，以提高真实感。

光照设置：

定义并添加DirectionalLight，以实现更逼真的光照效果。

用户交互：

实现鼠标控制（旋转、平移、缩放），使用户能够通过鼠标与模型互动。

配置鼠标行为（如MouseRotate、MouseZoom、MouseTranslate）以控制3D对象。

动画和行为：

实现螺旋桨的旋转动画，使模型看起来更加生动。

使用RotationInterpolator为螺旋桨添加持续旋转的效果。

实验考量

代码的组织与结构：确保代码结构清晰，易于理解和维护。

性能优化：确保3D渲染过程流畅，避免不必要的性能开销。

用户体验：关注用户与3D模型交互的流畅性和直观性。

总结

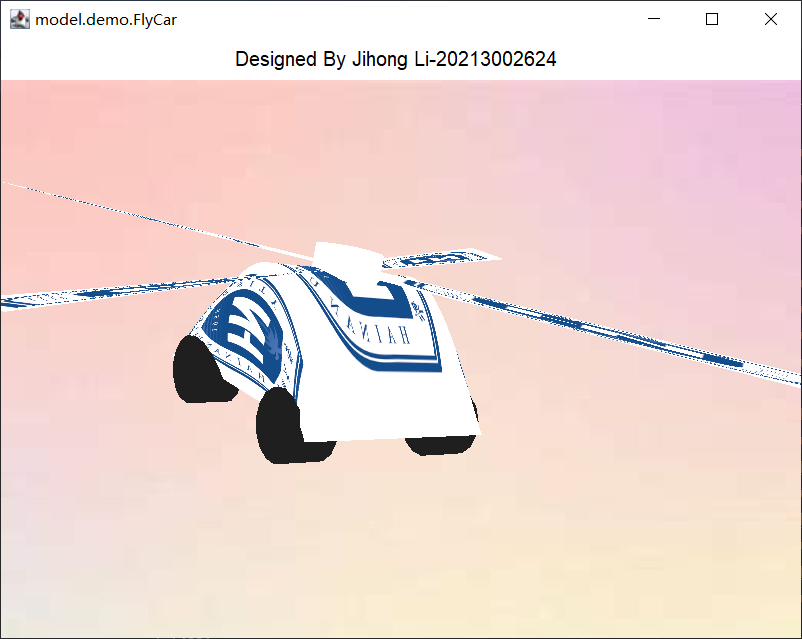
这个实验的核心在于利用Java 3D库来构建一个交互式的3D飞行汽车模型，其中包括对复杂3D形状的建模、纹理和材质的应用、光照效果的设置，以及用户交互功能的实现。通过这个项目，可以深入理解3D图形编程的核心概念，并掌握使用Java 3D API进行创意编程的技巧。

本实验中主要使用了Bezier三次曲面以及如何将双三次Bezier曲面的矩阵表示应用到java3D编程中。本次实验将Bezier曲线连接成曲面，然后设置高亮颜色，这是最关键的部分。并为曲面添加了纹理映射，将海南大学校徽映射到车身上面和旋转桨上面，并且添加了高亮反光，整个画面也添加了粉色渐变色背景便于观察效果图。

为了体现飞行效果，将飞机的旋转桨进行了旋转效果，运行代码会发现可以转动。车上身。

侧面两个车身，和车下面的底面，都是通过Bezier三次曲面，16个坐标点定位拼接而成，车轮通过坐标精确定位加入到汽车的四个方位，在顶部加入旋转桨中心轴，中心轴也是通过圆柱体坐标定位而成。

四个旋转桨也是Bezier三次曲面，通过精确坐标定位，把四个曲面“插”在中心轴中，形成“螺旋桨”，并进行分组旋转。如下图所示：



**二、实验过程**

## 实验内容：

1. 代码

package model.demo;  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseWheelZoom;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.geometry.Cylinder;  
import com.sun.j3d.utils.image.TextureLoader;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.\*;  
import java.applet.Applet;  
import java.awt.\*;  
import java.net.URL;  
  
public class FlyCar extends Applet {  
 public FlyCar() {  
 setLayout(new BorderLayout());  
 Panel p = new Panel();  
  
 // 创建一个新的 Font 对象  
 Font font = new Font("Dialog", Font.ROMAN\_BASELINE, 20); // 可以选择您喜欢的字体样式和大小  
  
 Label label = new Label("Designed By Jihong Li-20213002624");  
 label.setFont(font); // 将字体应用到 Label 上  
 p.add(label);  
  
 add(p, BorderLayout.NORTH);  
  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new FlyCar(), 800, 600);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
  
 // 定义背景  
 URL url = getClass().getClassLoader().getResource("model/demo/bg.png");  
 TextureLoader textureLoader = new TextureLoader(url, this);  
 ImageComponent2D image1 = textureLoader.getImage();  
  
 if (image1 == null) {  
 System.out.println("Image loading failed!");  
 } else {  
 System.out.println("Image loaded successfully!");  
 }  
  
  
 // 设置背景外观  
 Background bg = new Background();  
 bg.setImage(image1);  
 bg.setImageScaleMode(Background.SCALE\_FIT\_MAX); // 调整图像以填充整个背景  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
  
 Color3f directionalColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
  
  
 // 鼠标缩放部分----------------------------------------------------------------------------------------------------  
 Transform3D tr = new Transform3D();  
 tr.setScale(0.65);  
 //定义总的TransformGroup：transformgroup  
 TransformGroup transformgroup = new TransformGroup(tr);  
  
 //设置对该TransformGroup的读写能力  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 //将该TransformGroup加入到BranchGroupRoot中  
 BranchGroupRoot.addChild(transformgroup);  
  
 //定义鼠标对场景的旋转、平移与放大功能  
 //鼠标旋转功能  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
  
 //这是之前的不能进行缩放  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);

//重新定义鼠标缩放功能默认滚轮  
 MouseWheelZoom mouseWheelZoom = new MouseWheelZoom();  
 mouseWheelZoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouseWheelZoom);  
 mouseWheelZoom.setSchedulingBounds(bounds);  
  
 //鼠标平移功能默认鼠标右键  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 // 鼠标缩放部分----------------------------------------------------------------------------------------------------  
  
 //定义外观app1  
 Appearance app1 = new Appearance();  
  
  
 // 创建 ColoringAttributes 并设置颜色  
 ColoringAttributes newColoringAttributes = new ColoringAttributes();  
 newColoringAttributes.setColor(new Color3f(0.0f, 0.0f, 1.0f)); // 设置为蓝色  
  
 // 创建 Material 并设置颜色  
 Material newMaterial = new Material();  
 newMaterial.setDiffuseColor(new Color3f(0.0f, 0.0f, 1.0f)); // 设置漫反射颜色，即蓝色  
  
 // 应用新的 ColoringAttributes 和 Material 到外观 app1  
 app1.setColoringAttributes(newColoringAttributes);  
 app1.setMaterial(newMaterial);  
  
 // 加载纹理图片  
 TextureLoader loader = new TextureLoader("src/model/demo/xiaohui.png", 2, this);  
 ImageComponent2D image = loader.getImage();  
 Texture2D texture = new Texture2D(Texture.BASE\_LEVEL, Texture.RGBA,  
 image.getWidth(), image.getHeight());  
 texture.setImage(0, image);  
 texture.setEnable(true);  
//设置纹理的放大与缩小过滤  
 texture.setMagFilter(Texture.BASE\_LEVEL\_LINEAR);  
 texture.setMinFilter(Texture.BASE\_LEVEL\_LINEAR);  
  
  
 app1.setTexture(texture);  
  
 PolygonAttributes polygona1 = new PolygonAttributes();  
 polygona1.setBackFaceNormalFlip(true);  
 polygona1.setCullFace(PolygonAttributes.CULL\_NONE);  
 app1.setPolygonAttributes(polygona1);  
 app1.setCapability(Appearance.ALLOW\_TEXGEN\_WRITE);  
  
 //定义外观app2

Appearance app2 = new Appearance();  
 PolygonAttributes polygona2 = new PolygonAttributes();  
 polygona2.setBackFaceNormalFlip(true);  
 polygona2.setCullFace(PolygonAttributes.CULL\_NONE);  
 app2.setPolygonAttributes(polygona2);  
 app2.setCapability(Appearance.ALLOW\_TEXGEN\_WRITE);  
// 加载纹理图片  
 TextureLoader loader2 = new TextureLoader("src/model/demo/wheel.png", 2, this);  
 ImageComponent2D image2 = loader2.getImage();  
 Texture2D texture2 = new Texture2D(Texture.BASE\_LEVEL, Texture.RGBA,  
 image2.getWidth(), image2.getHeight());  
 texture2.setImage(0, image2);  
 texture2.setEnable(true);  
//设置纹理的放大与缩小过滤  
 texture2.setMagFilter(Texture.BASE\_LEVEL\_LINEAR);  
 texture2.setMinFilter(Texture.BASE\_LEVEL\_LINEAR);  
 app2.setTexture(texture2);  
 // 创建 ColoringAttributes 并设置颜色  
 ColoringAttributes newColoringAttributes2 = new ColoringAttributes();  
 newColoringAttributes2.setColor(new Color3f(0.0f, 1.0f, 0.0f)); // 设置为黑色  
  
 // 创建 Material 并设置颜色  
 Material newMaterial2 = new Material();  
 newMaterial2.setDiffuseColor(new Color3f(0.0f, 1.0f, 0.0f)); // 设置漫反射颜色，即黑色  
  
 // 应用新的 ColoringAttributes 和 Material 到外观 app1  
 app2.setColoringAttributes(newColoringAttributes2);  
 app2.setMaterial(newMaterial2);  
  
 //定义外观app3  
 Appearance app3 = new Appearance();  
 PolygonAttributes polygona3 = new PolygonAttributes();  
 polygona3.setBackFaceNormalFlip(true);  
 polygona3.setCullFace(PolygonAttributes.CULL\_NONE);  
 app3.setPolygonAttributes(polygona3);  
 app3.setCapability(Appearance.ALLOW\_TEXGEN\_WRITE);  
 // 加载纹理图片  
 TextureLoader loader3 = new TextureLoader("src/model/demo/xiaohui.png", 2, this);  
 ImageComponent2D image3 = loader3.getImage();  
 Texture2D texture3 = new Texture2D(Texture.BASE\_LEVEL, Texture.RGBA,  
 image3.getWidth(), image3.getHeight());  
 texture3.setImage(0, image3);  
 texture3.setEnable(true);  
//设置纹理的放大与缩小过滤  
 texture3.setMagFilter(Texture.BASE\_LEVEL\_LINEAR);  
 texture3.setMinFilter(Texture.BASE\_LEVEL\_LINEAR);  
 app3.setTexture(texture3);  
 // 创建 ColoringAttributes 并设置颜色  
 ColoringAttributes newColoringAttributes3 = new ColoringAttributes();  
 newColoringAttributes3.setColor(new Color3f(1.0f, 0.0f, 0.0f)); // 设置为红色

// 创建 Material 并设置颜色  
Material newMaterial3 = new Material();  
newMaterial3.setDiffuseColor(new Color3f(1.0f, 0.0f, 0.0f)); // 设置漫反射颜色，即红色  
  
// 应用新的 ColoringAttributes 和 Material 到外观 app1  
app3.setColoringAttributes(newColoringAttributes3);  
app3.setMaterial(newMaterial3);  
  
//定义控制顶点  
float[][][] back = {  
 {{-0.3f, 0.f, 0.1f, 1.f},  
 {-0.2f, 0.f, 0.1f, 1.f},  
 {-0.1f, 0.f, 0.1f, 1.f},  
 {-0.f, -0.f, 0.1f, 1.f}},  
  
 {{-0.3f, 0.f, 0.05f, 1.f},  
 {-0.2f, 0.f, 0.05f, 1.f},  
 {-0.1f, 0.f, 0.05f, 1.f},  
 {-0.f, -0.f, 0.05f, 1.f}},  
  
  
 {{-0.3f, 0.f, -0.05f, 1.f},  
 {-0.2f, 0.f, -0.05f, 1.f},  
 {-0.1f, 0.f, -0.05f, 1.f},  
 {-0.f, -0.f, -0.05f, 1.f}},  
  
 {{-0.3f, 0.f, -0.1f, 1.f},  
 {-0.2f, 0.f, -0.1f, 1.f},  
 {-0.1f, 0.f, -0.1f, 1.f},  
 {-0.f, -0.f, -0.1f, 1.f}}};  
  
  
float[][][] body = {  
 {{-0.3f, 0.f, 0.1f, 1.f},  
 {-0.3f, 0.f, 0.05f, 1.f},  
 {-0.3f, 0.f, -0.05f, 1.f},  
 {-0.3f, -0.f, -0.1f, 1.f}},  
  
 {{-0.2f, 0.2f, 0.1f, 1.f},  
 {-0.2f, 0.2f, 0.05f, 1.f},  
 {-0.2f, 0.2f, -0.05f, 1.f},  
 {-0.2f, 0.2f, -0.1f, 1.f}},  
  
  
 {{-0.1f, 0.2f, 0.1f, 1.f},  
 {-0.1f, 0.2f, 0.05f, 1.f},  
 {-0.1f, 0.2f, -0.05f, 1.f},  
 {-0.1f, 0.2f, -0.1f, 1.f}},

{{-0.f, 0.f, 0.1f, 1.f},  
 {-0.f, 0.f, 0.05f, 1.f},  
 {-0.f, 0.f, -0.05f, 1.f},  
 {-0.f, -0.f, -0.1f, 1.f}}};  
  
float[][][] ceshen1 = {  
 {{-0.3f, 0.f, 0.1f, 1.f},  
 {-0.2f, 0.f, 0.1f, 1.f},  
 {-0.1f, 0.f, 0.1f, 1.f},  
 {-0.f, -0.f, 0.1f, 1.f}},  
  
  
 {{-0.3f, 0.f, 0.13f, 1.f},  
 {-0.2f, 0.08f, 0.13f, 1.f},  
 {-0.1f, 0.08f, 0.13f, 1.f},  
 {0.f, 0.f, 0.13f, 1.f}},  
  
  
 {{-0.3f, 0.f, 0.13f, 1.f},  
 {-0.2f, 0.15f, 0.13f, 1.f},  
 {-0.1f, 0.15f, 0.13f, 1.f},  
 {0.f, 0.f, 0.13f, 1.f}},  
  
 {{-0.3f, 0.f, 0.1f, 1.f},  
 {-0.2f, 0.2f, 0.1f, 1.f},  
 {-0.1f, 0.2f, 0.1f, 1.f},  
 {0.f, 0.f, 0.1f, 1.f}},  
};  
  
float[][][] ceshen2 = {  
 {{-0.3f, 0.f, -0.1f, 1.f},  
 {-0.2f, 0.f, -0.1f, 1.f},  
 {-0.1f, 0.f, -0.1f, 1.f},  
 {-0.f, -0.f, -0.1f, 1.f}},  
  
  
 {{-0.3f, 0.f, -0.13f, 1.f},  
 {-0.2f, 0.08f, -0.13f, 1.f},  
 {-0.1f, 0.08f, -0.13f, 1.f},  
 {0.f, 0.f, -0.13f, 1.f}},  
  
  
 {{-0.3f, 0.f, -0.13f, 1.f},  
 {-0.2f, 0.15f, -0.13f, 1.f},  
 {-0.1f, 0.15f, -0.13f, 1.f},  
 {0.f, 0.f, -0.13f, 1.f}},  
  
 {{-0.3f, 0.f, -0.1f, 1.f},  
 {-0.2f, 0.2f, -0.1f, 1.f},  
 {-0.1f, 0.2f, -0.1f, 1.f},  
 {0.f, 0.f, -0.1f, 1.f}},  
};

//螺旋桨1  
float[][][] P1 = {  
 {{-0.16f, 0.15f, 0.03f, 1.f},  
 {-0.153f, 0.1523f, 0.03f, 1.f},  
 {-0.147f, 0.1547f, 0.03f, 1.f},  
 {-0.14f, 0.157f, 0.03f, 1.f}},  
  
 {{-0.167f, 0.15f, 0.23f, 1.f},  
 {-0.152f, 0.1523f, 0.23f, 1.f},  
 {-0.148f, 0.1547f, 0.23f, 1.f},  
 {-0.133f, 0.157f, 0.23f, 1.f}},  
  
 {{-0.173f, 0.15f, 0.43f, 1.f},  
 {-0.158f, 0.1523f, 0.43f, 1.f},  
 {-0.142f, 0.1547f, 0.43f, 1.f},  
 {-0.127f, 0.157f, 0.43f, 1.f}},  
  
 {{-0.18f, 0.15f, 0.63f, 1.f},  
 {-0.163f, 0.1523f, 0.63f, 1.f},  
 {-0.14f, 0.1547f, 0.63f, 1.f},  
 {-0.12f, 0.157f, 0.63f, 1.f}}  
};  
  
  
//螺旋桨2  
float[][][] P2 = {  
 {{-0.16f, 0.157f, -0.03f, 1.f},  
 {-0.153f, 0.1547f, -0.03f, 1.f},  
 {-0.147f, 0.1523f, -0.03f, 1.f},  
 {-0.14f, 0.15f, -0.03f, 1.f}},  
  
 {{-0.167f, 0.157f, -0.23f, 1.f},  
 {-0.152f, 0.1547f, -0.23f, 1.f},  
 {-0.148f, 0.1523f, -0.23f, 1.f},  
 {-0.133f, 0.15f, -0.23f, 1.f}},  
  
 {{-0.173f, 0.157f, -0.43f, 1.f},  
 {-0.158f, 0.1547f, -0.43f, 1.f},  
 {-0.142f, 0.1523f, -0.43f, 1.f},  
 {-0.127f, 0.15f, -0.43f, 1.f}},  
  
 {{-0.18f, 0.157f, -0.63f, 1.f},  
 {-0.163f, 0.1547f, -0.63f, 1.f},  
 {-0.14f, 0.1523f, -0.63f, 1.f},  
 {-0.12f, 0.15f, -0.63f, 1.f}}  
};

//螺旋桨3  
float[][][] P3 = {  
 {  
 {-0.15f, 0.15f, 0.01f, 1.0f},  
 {-0.15f, 0.1523f, 0.003f, 1.0f},  
 {-0.15f, 0.1547f, -0.003f, 1.0f},  
 {-0.15f, 0.157f, -0.01f, 1.0f}  
 },  
 {  
 {0.19f, 0.15f, 0.016f, 1.0f},  
 {0.19f, 0.1523f, 0.016f, 1.0f},  
 {0.19f, 0.1547f, -0.016f, 1.0f},  
 {0.19f, 0.157f, -0.016f, 1.0f}  
 },  
 {  
 {0.39f, 0.15f, 0.023f, 1.0f},  
 {0.39f, 0.1523f, 0.023f, 1.0f},  
 {0.39f, 0.1547f, -0.023f, 1.0f},  
 {0.39f, 0.157f, -0.023f, 1.0f}  
 },  
 {  
 {0.45f, 0.15f, 0.03f, 1.0f},  
 {0.45f, 0.1523f, 0.01f, 1.0f},  
 {0.45f, 0.1547f, -0.01f, 1.0f},  
 {0.45f, 0.157f, -0.03f, 1.0f}  
 }  
};  
  
  
//螺旋桨4  
float[][][] P4 = {  
 {  
 {-0.15f, 0.157f, 0.01f, 1.0f},  
 {-0.15f, 0.1547f, 0.003f, 1.0f},  
 {-0.15f, 0.1523f, -0.003f, 1.0f},  
 {-0.15f, 0.15f, -0.01f, 1.0f}  
 },  
 {  
 {-0.35f, 0.157f, 0.016f, 1.0f},  
 {-0.35f, 0.1547f, 0.016f, 1.0f},  
 {-0.35f, 0.1523f, -0.016f, 1.0f},  
 {-0.35f, 0.15f, -0.016f, 1.0f}  
 },  
 {  
 {-0.55f, 0.157f, 0.023f, 1.0f},  
 {-0.55f, 0.1547f, 0.023f, 1.0f},  
 {-0.55f, 0.1523f, -0.023f, 1.0f},  
 {-0.55f, 0.15f, -0.023f, 1.0f}  
 },

{  
 {-0.75f, 0.157f, 0.03f, 1.0f},  
 {-0.75f, 0.1547f, 0.01f, 1.0f},  
 {-0.75f, 0.1523f, -0.01f, 1.0f},  
 {-0.75f, 0.15f, -0.03f, 1.0f}  
 }  
};  
  
  
// 创建车底曲面  
Shape3D carTopPart = new BezierThreeOrderSurfaceface(back, app1);  
transformgroup.addChild(carTopPart);  
  
//创建车身曲面  
Shape3D carWheelPart = new BezierThreeOrderSurfaceface(body, app1);  
transformgroup.addChild(carWheelPart);  
  
//创建侧车身曲面1  
Shape3D carceshen1 = new BezierThreeOrderSurfaceface(ceshen1, app1);  
transformgroup.addChild(carceshen1);  
  
//创建侧车身曲面1  
Shape3D carceshen2 = new BezierThreeOrderSurfaceface(ceshen2, app1);  
transformgroup.addChild(carceshen2);  
  
Shape3D luo1 = new BezierThreeOrderSurfaceface(P1, app1);  
  
Shape3D luo2 = new BezierThreeOrderSurfaceface(P2, app1);  
  
Shape3D luo3 = new BezierThreeOrderSurfaceface(P3, app1);  
  
Shape3D luo4 = new BezierThreeOrderSurfaceface(P4, app1);  
  
//螺旋桨组  
TransformGroup tg = new TransformGroup();  
tg.addChild(luo1);  
tg.addChild(luo2);  
tg.addChild(luo3);  
tg.addChild(luo4);  
tg.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
tg.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
transformgroup.addChild(tg);  
  
//螺旋桨中心  
Transform3D t = new Transform3D();  
t.setTranslation(new Vector3f(-0.15f, 0.15f, 0.f));  
TransformGroup tg1 = new TransformGroup(t);  
tg1.addChild(new Cylinder(0.04f, 0.05f, app3));  
  
transformgroup.addChild(tg1);

//四个轮子  
 t = new Transform3D();  
 t.rotX(Math.PI / 2);  
 t.setTranslation(new Vector3f(-0.25f, 0.f, 0.1f));  
 TransformGroup tg2 = new TransformGroup(t);  
 tg2.addChild(new Cylinder(0.04f, 0.05f, app2));  
 transformgroup.addChild(tg2);  
  
 t = new Transform3D();  
 t.rotX(Math.PI / 2);  
 t.setTranslation(new Vector3f(-0.05f, 0.f, 0.1f));  
 TransformGroup tg3 = new TransformGroup(t);  
 tg3.addChild(new Cylinder(0.04f, 0.05f, app2));  
 transformgroup.addChild(tg3);  
  
 t = new Transform3D();  
 t.rotX(Math.PI / 2);  
 t.setTranslation(new Vector3f(-0.25f, 0.f, -0.1f));  
 TransformGroup tg4 = new TransformGroup(t);  
 tg4.addChild(new Cylinder(0.04f, 0.05f, app2));  
 transformgroup.addChild(tg4);  
  
 t = new Transform3D();  
 t.rotX(Math.PI / 2);  
 t.setTranslation(new Vector3f(-0.05f, 0.f, -0.1f));  
 TransformGroup tg5 = new TransformGroup(t);  
 tg5.addChild(new Cylinder(0.04f, 0.05f, app2));  
 transformgroup.addChild(tg5);  
  
 //螺旋桨旋转  
 Alpha rotationAlpha1 = new Alpha(-1, Alpha.INCREASING\_ENABLE, 0, 0, 2000, 0, 0, 0, 0, 0);  
 Transform3D zAxis1 = new Transform3D();  
 zAxis1.setTranslation(new Vector3f(-0.15f, 0.15f, 0.f));  
 RotationInterpolator rotator1 = new RotationInterpolator(rotationAlpha1, tg, zAxis1, 0.0f, (float) Math.PI \* 2.0f);  
 rotator1.setSchedulingBounds(bounds);  
 transformgroup.addChild(rotator1);  
  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}

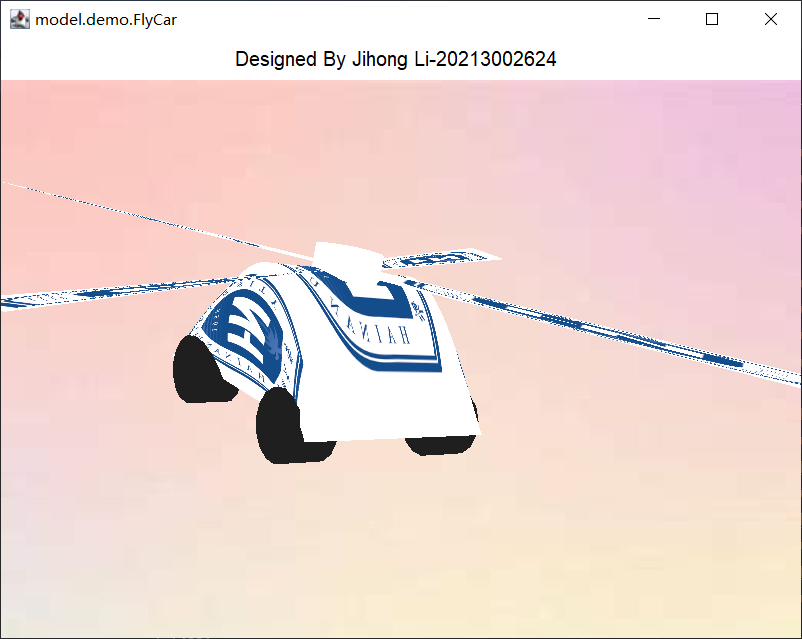
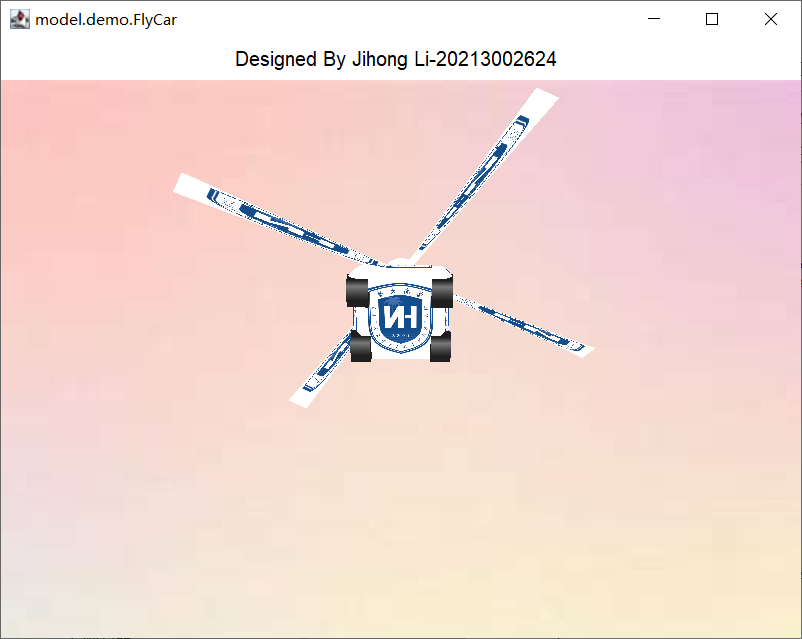
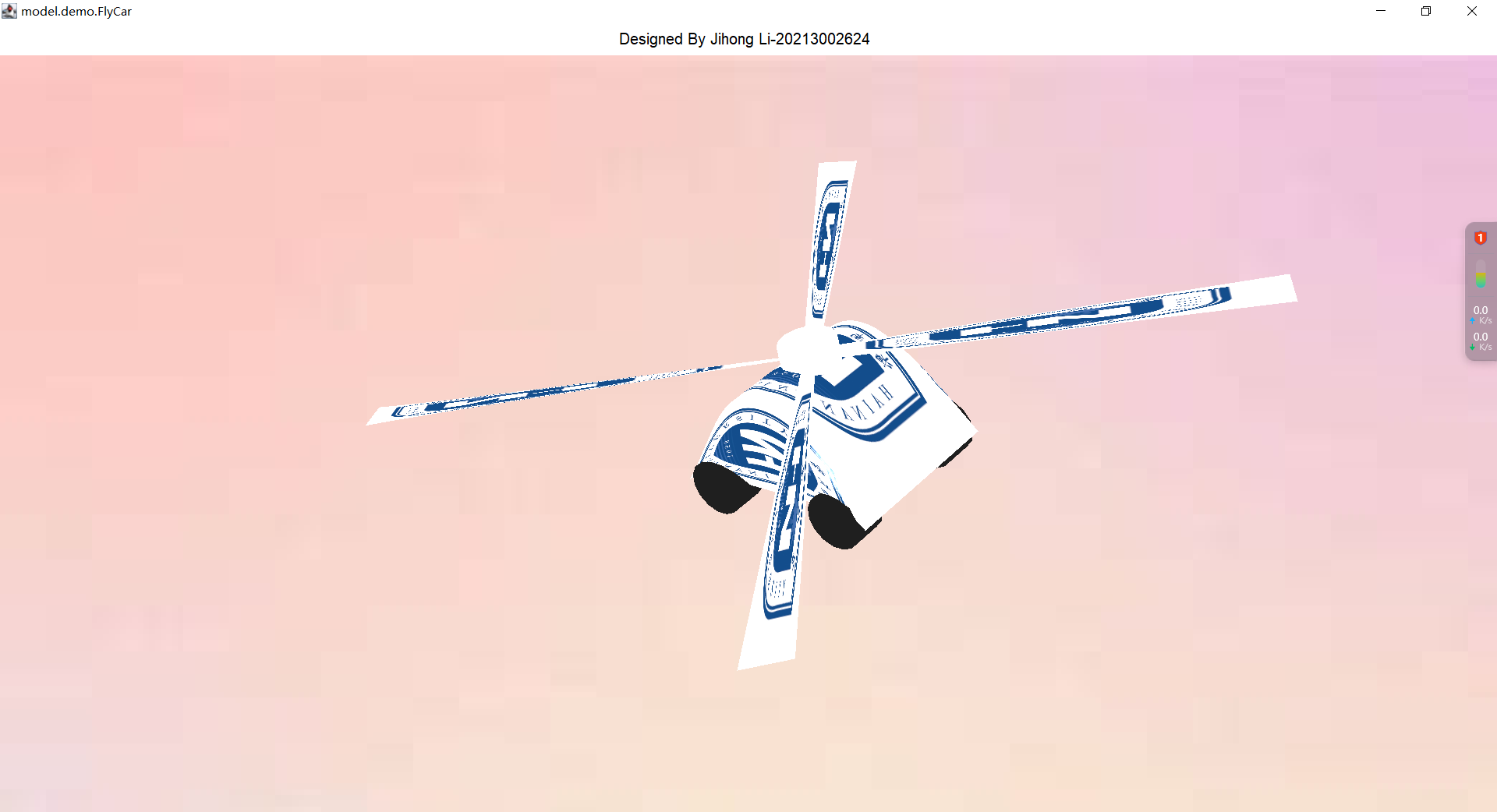
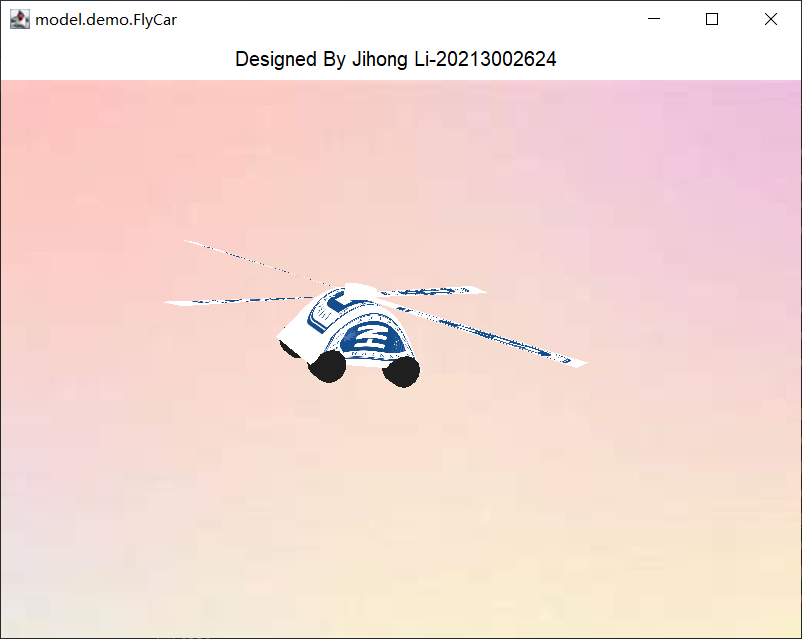
class BezierThreeOrderSurfaceface extends Shape3D {  
 public BezierThreeOrderSurfaceface(float[][][] P, Appearance app) {  
 int i, j, k;  
 int n0;//定义对参数u、v在[0，1]区间的等分点数  
 float division;//参数u在[0，1]区间的等分线段长度  
 n0 = 50;  
 division = 1.f / n0;  
//分别定义存放控制顶点x、y、z坐标与第四维坐标的数组  
 float[][] PX = new float[4][4];  
 float[][] PY = new float[4][4];  
 float[][] PZ = new float[4][4];  
 float[][] P4 = new float[4][4];  
//定义系数矩阵及其转置矩阵  
 float[][] M1 = {{1.f, 0.f, 0.f, 0.f},  
 {-3.f, 3.f, 0.f, 0.f},  
 {3.f, -6.f, 3.f, 0.f},  
 {-1.f, 3.f, -3.f, 1.f}};  
 float[][] M2 = {{1.f, -3.f, 3.f, -1.f},  
 {0.f, 3.f, -6.f, 3.f},  
 {0.f, 0.f, 3.f, -3.f},  
 {0.f, 0.f, 0.f, 1.f}};  
//定义Bezier曲面的u、v参数分割点坐标数组  
 float[][][] UV = new float[n0 + 1][n0 + 1][2];  
//定义U、V矩阵数组  
 float[][] UU = new float[1][4];  
 float[][] VV = new float[4][1];  
//定义存放曲面上点的坐标的数组  
 float[][][] SurfaceXYZ = new float[n0 + 1][n0 + 1][4];  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UV[i][j][0] = i \* division;  
 UV[i][j][1] = j \* division;  
 }  
 for (i = 0; i < 4; i++)  
 for (j = 0; j < 4; j++) {  
 PX[i][j] = P[i][j][0];  
 PY[i][j] = P[i][j][1];  
 PZ[i][j] = P[i][j][2];  
 P4[i][j] = P[i][j][3];  
 }

//计算曲面上点的坐标值  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UU[0][0] = 1.f;  
 UU[0][1] = UV[i][j][0];  
 UU[0][2] = UV[i][j][0] \* UV[i][j][0];  
 UU[0][3] = UV[i][j][0] \* UV[i][j][0] \* UV[i][j][0];  
 VV[0][0] = 1.f;  
 VV[1][0] = UV[i][j][1];  
 VV[2][0] = UV[i][j][1] \* UV[i][j][1];  
 VV[3][0] = UV[i][j][1] \* UV[i][j][1] \* UV[i][j][1];  
 //计算一点的x坐标  
 matrixm g0 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g1 = new matrixm(1, 4, 4, g0.CC, PX);  
 matrixm g2 = new matrixm(1, 4, 4, g1.CC, M2);  
 matrixm g3 = new matrixm(1, 4, 1, g2.CC, VV);  
 SurfaceXYZ[i][j][0] = g3.CC[0][0];  
 //计算一点的y坐标  
 matrixm g4 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g5 = new matrixm(1, 4, 4, g4.CC, PY);  
 matrixm g6 = new matrixm(1, 4, 4, g5.CC, M2);  
 matrixm g7 = new matrixm(1, 4, 1, g6.CC, VV);  
 SurfaceXYZ[i][j][1] = g7.CC[0][0];  
 //计算一点的z坐标  
 matrixm g8 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g9 = new matrixm(1, 4, 4, g8.CC, PZ);  
 matrixm g10 = new matrixm(1, 4, 4, g9.CC, M2);  
 matrixm g11 = new matrixm(1, 4, 1, g10.CC, VV);  
 SurfaceXYZ[i][j][2] = g11.CC[0][0];  
 //计算一点的第4维坐标  
 matrixm g12 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g13 = new matrixm(1, 4, 4, g12.CC, P4);  
 matrixm g14 = new matrixm(1, 4, 4, g13.CC, M2);  
 matrixm g15 = new matrixm(1, 4, 1, g14.CC, VV);  
 SurfaceXYZ[i][j][3] = g15.CC[0][0];  
 //将齐次坐标转换为三维坐标系坐标，如果第4维为1，则不用除第4维  
 SurfaceXYZ[i][j][0] = SurfaceXYZ[i][j][0] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][1] = SurfaceXYZ[i][j][1] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][2] = SurfaceXYZ[i][j][2] / SurfaceXYZ[i][j][3];  
 }  
 QuadArray BezierQuadsurfaceface = new QuadArray(n0 \* n0 \* 4, GeometryArray.COORDINATES  
 | GeometryArray.NORMALS | GeometryArray.TEXTURE\_COORDINATE\_2);  
 int c = 0;//以顶点数累加的方式设置顶点的序号  
 for (i = 0; i < n0; i++) {  
 for (j = 0; j < n0; j++) {//设置一个平面上的4个点  
 Point3f A = new Point3f(SurfaceXYZ[i][j][0],  
 SurfaceXYZ[i][j][1], SurfaceXYZ[i][j][2]);  
 Point3f B = new Point3f(SurfaceXYZ[i + 1][j][0],  
 SurfaceXYZ[i + 1][j][1], SurfaceXYZ[i + 1][j][2]);  
 Point3f C = new Point3f(SurfaceXYZ[i + 1][j + 1][0],  
 SurfaceXYZ[i + 1][j + 1][1], SurfaceXYZ[i + 1][j + 1][2]);  
 Point3f D = new Point3f(SurfaceXYZ[i][j + 1][0],  
 SurfaceXYZ[i][j + 1][1], SurfaceXYZ[i][j + 1][2]);  
//计算由四个点形成的平面的法向量  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
//设置点的序号

BezierQuadsurfaceface.setCoordinate(c, A);  
 BezierQuadsurfaceface.setCoordinate(c + 1, B);  
 BezierQuadsurfaceface.setCoordinate(c + 2, C);  
 BezierQuadsurfaceface.setCoordinate(c + 3, D);  
//设置点的法向量  
 BezierQuadsurfaceface.setNormal(c, n);  
 BezierQuadsurfaceface.setNormal(c + 1, n);  
 BezierQuadsurfaceface.setNormal(c + 2, n);  
 BezierQuadsurfaceface.setNormal(c + 3, n);  
//设置纹理坐标  
 TexCoord2f texCoords = new TexCoord2f(i \* 1.f / n0, 1.f - j \* 1.f / n0);  
 BezierQuadsurfaceface.setTextureCoordinate(0, c, texCoords);  
 texCoords = new TexCoord2f((i + 1) \* 1.f / n0, 1.f - j \* 1.f / n0);  
 BezierQuadsurfaceface.setTextureCoordinate(0, c + 1, texCoords);  
 texCoords = new TexCoord2f((i + 1) \* 1.f / n0, 1.f - (j + 1) \* 1.f / n0);  
 BezierQuadsurfaceface.setTextureCoordinate(0, c + 2, texCoords);  
 texCoords = new TexCoord2f(i \* 1.f / n0, 1.f - (j + 1) \* 1.f / n0);  
 BezierQuadsurfaceface.setTextureCoordinate(0, c + 3, texCoords);  
 c = c + 4;  
 }  
 }  
 this.addGeometry(BezierQuadsurfaceface);  
 this.setAppearance(app);  
 }  
}

class matrixm {  
 public float CC[][] = new float[4][4];  
 int ll, mm, kk;  
  
 public matrixm(int mmm, int kkk, int nnn, float a[][], float b[][]) {  
 for (ll = 0; ll < mmm; ll++)  
 for (mm = 0; mm < nnn; mm++) {  
 CC[ll][mm] = 0.f;  
 }  
 for (ll = 0; ll < mmm; ll++)  
 for (mm = 0; mm < nnn; mm++) {  
 for (kk = 0; kk < kkk; kk++) CC[ll][mm] = CC[ll][mm] + a[ll][kk] \* b[kk][mm];  
 }  
 }  
}

1. 运行效果



**《计算机图形学》实验报告一**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.9.14  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

1.了解并熟练掌握java3D编程

2.了解Java3D基本图形功能

3.了解并熟悉Java3D的场景式管理

4.通过球体和长方体组合体程序对Java场景式管理实践

**二、实验过程**

## 实验内容一：

（1）代码

1. *//加入程序运行时需要的Java、Java 3D的包*
2. import com.sun.j3d.utils.applet.MainFrame;
3. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
4. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
5. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
6. import com.sun.j3d.utils.geometry.Box;
7. import com.sun.j3d.utils.geometry.Sphere;
8. import com.sun.j3d.utils.universe.SimpleUniverse;
9. import javax.media.j3d.\*;
10. import javax.vecmath.Color3f;
11. import javax.vecmath.Point3d;
12. import javax.vecmath.Vector3f;
13. import java.applet.Applet;
14. import java.awt.\*;
15. public class twoprimitivedisplay extends Applet {
16. public twoprimitivedisplay() {*//设置显示界面的相关参数*
17. setLayout(new BorderLayout());
18. *//创建投影平面Canvas3D*
19. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
20. Canvas3D c = new Canvas3D(gc);
21. *//将投影平面上的图象显示在显示平面的中间*
22. add("Center", c);
23. *//设置SimpleUniverse，由系统选择视点在z轴的正向，观察方向沿z轴反向*
24. BranchGroup BranchGroupScene = createBranchGroup();
25. SimpleUniverse u = new SimpleUniverse(c);
26. u.getViewingPlatform().setNominalViewingTransform();
27. *//将BranchGroup：BranchGroupScene加入到SimpleUniverse：u中*
28. u.addBranchGraph(BranchGroupScene);
29. }
30. public static void main(String[] args) {*//通过MainFrame显示图象*
31. new MainFrame(new twoprimitivedisplay(), 300, 300);
32. }
33. public BranchGroup createBranchGroup() {*//定义BranchGroup*
34. BranchGroup BranchGroupRoot = new BranchGroup();
35. *//创建球心在坐标系原点球形范围*
36. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
37. *//定义背景颜色*
38. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
39. Background bg = new Background(bgColor);
40. bg.setApplicationBounds(bounds);
41. BranchGroupRoot.addChild(bg);
42. *//定义平行光、颜色、照射方向与作用范围*
43. Color3f directionalColor = new Color3f(1.f, 1.f, 1.f);
44. Vector3f vec = new Vector3f(-1.f, -1.f, -1.0f);
45. DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);
46. directionalLight.setInfluencingBounds(bounds);
47. BranchGroupRoot.addChild(directionalLight);
48. *//定义两个三维型体的外观*
49. Appearance app1 = new Appearance();
50. Material material1 = new Material();
51. material1.setDiffuseColor(new Color3f(1.0f, .0f, 0.0f));
52. app1.setMaterial(material1);
53. Appearance app2 = new Appearance();
54. Material material2 = new Material();
55. material2.setDiffuseColor(new Color3f(.0f, 1.0f, 0.0f));
56. app2.setMaterial(material2);
57. *//定义总的TransformGroup：transformgroup*
58. TransformGroup transformgroup = new TransformGroup();
59. *//设置对该TransformGroup的读写能力*
60. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
61. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
62. *//将该TransformGroup加入到BranchGroupRoot中*
63. BranchGroupRoot.addChild(transformgroup);
64. *//定义鼠标对场景的旋转、平移与放大功能*
65. MouseRotate mouserotate = new MouseRotate();
66. mouserotate.setTransformGroup(transformgroup);
67. BranchGroupRoot.addChild(mouserotate);
68. mouserotate.setSchedulingBounds(bounds);
69. MouseZoom mousezoom = new MouseZoom();
70. mousezoom.setTransformGroup(transformgroup);
71. BranchGroupRoot.addChild(mousezoom);
72. mousezoom.setSchedulingBounds(bounds);
73. MouseTranslate mousetranslate = new MouseTranslate();
74. mousetranslate.setTransformGroup(transformgroup);
75. BranchGroupRoot.addChild(mousetranslate);
76. mousetranslate.setSchedulingBounds(bounds);
77. */\*定义一个球体与一个长方体的大小、外观属性与坐标变换，并定义相应的TransformGroup：tg1、tg2\*/*
78. TransformGroup tg1 = new TransformGroup();
79. tg1.addChild(new Sphere(0.4f, app1));
80. Transform3D t = new Transform3D();
81. t.setTranslation(new Vector3f(0.f, -0.425f, 0.f));
82. TransformGroup tg2 = new TransformGroup(t);
83. tg2.addChild(new Box(0.5f, 0.05f, 0.5f, app2));
84. *//将定义好的两个TransformGroup(tg1、tg2)加入到总的transformgroup*
85. transformgroup.addChild(tg1);
86. transformgroup.addChild(tg2);
87. *//对BranchGroupRoot预编译*
88. BranchGroupRoot.compile();
89. *//通过方法名返回BranchGroupRoot*
90. return BranchGroupRoot;
91. }
92. }

（2）结果截图

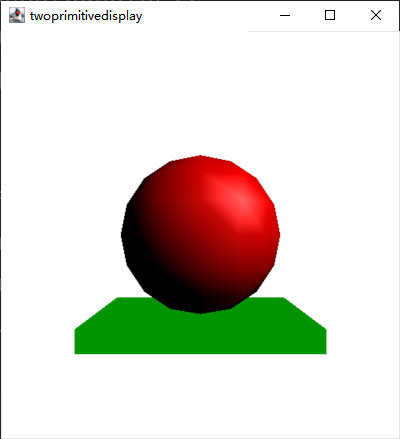


图1- 1 运行结果截图

## 实验内容二：

在上述基础上修改了代码：

（一）代码：

1. import com.sun.j3d.utils.applet.MainFrame;
2. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
3. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
4. import com.sun.j3d.utils.behaviors.mouse.MouseWheelZoom;
5. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
6. import com.sun.j3d.utils.geometry.Box;
7. import com.sun.j3d.utils.geometry.Sphere;
8. import com.sun.j3d.utils.universe.SimpleUniverse;
9. import javax.media.j3d.\*;
10. import javax.vecmath.Color3f;
11. import javax.vecmath.Point3d;
12. import javax.vecmath.Vector3f;
13. import java.applet.Applet;
14. import java.awt.\*;
15. import java.awt.event.MouseEvent;
16. public class twoprimitivedisplay extends Applet {   *//第一次计算机图形学实验 2023-09-14晚上9-10节课*
17. public twoprimitivedisplay() {*//设置显示界面的相关参数*
18. setLayout(new BorderLayout());*//创建投影平面Canvas3D*
19. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
20. Canvas3D c = new Canvas3D(gc);
21. *//将投影平面上的图象显示在显示平面的中间*
22. add("Center", c);
23. *//设置SimpleUniverse，由系统选择视点在z轴的正向，观察方向沿z轴反向*
24. BranchGroup BranchGroupScene = createBranchGroup();
25. SimpleUniverse u = new SimpleUniverse(c);
26. u.getViewingPlatform().setNominalViewingTransform();*//将BranchGroup：BranchGroupScene加入到SimpleUniverse：u中*
27. u.addBranchGraph(BranchGroupScene);
28. }
29. public static void main(String[] args) {*//通过MainFrame显示图象*
30. new MainFrame(new twoprimitivedisplay(), 400, 400);
31. }
32. public BranchGroup createBranchGroup() {*//定义BranchGroup*
33. BranchGroup BranchGroupRoot = new BranchGroup();
34. *//        TransformGroup sceneTG=null;*
35. *//        BranchGroup scenceBranchGroupRoot =null;*
36. *//创建球心在坐标系原点球形范围*
37. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
38. *//定义背景颜色*
39. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
40. Background bg = new Background(bgColor);
41. bg.setApplicationBounds(bounds);
42. BranchGroupRoot.addChild(bg);
43. *//定义平行光、颜色、照射方向与作用范围*
44. Color3f directionalColor = new Color3f(1.f, 1.f, 1.f);
45. Vector3f vec = new Vector3f(-1.f, -1.f, -1.0f);
46. DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);
47. directionalLight.setInfluencingBounds(bounds);
48. BranchGroupRoot.addChild(directionalLight);
49. *//定义两个三维型体的外观*
50. *//        Appearance app1 = new Appearance();*
51. *//        Material material1 = new Material();*
52. *//        material1.setDiffuseColor(new Color3f(0.0f, .0f, 1.0f));*
53. *//        app1.setMaterial(material1);*
54. *// 创建具有光滑表面属性的 Appearance*
55. Appearance app1 = new Appearance();
56. *// 设置高光反射属性*
57. Material material1 = new Material();
58. material1.setDiffuseColor(new Color3f(0.0f, 0.0f, 1.0f)); *// 设置漫反射颜色*
59. material1.setSpecularColor(new Color3f(1.0f, 1.0f, 1.0f)); *// 设置高光反射颜色*
60. material1.setShininess(2500.0f); *// 设置光滑度，值越高越光滑*
61. app1.setMaterial(material1);
62. *//        Appearance app2 = new Appearance();*
63. *//        Material material2 = new Material();*
64. *//        material2.setDiffuseColor(new Color3f(.0f, 0.0f, 0.0f));*
65. *//        app2.setMaterial(material2);*
66. *// 创建具有透明度的 Appearance*
67. Appearance app2 = new Appearance();
68. TransparencyAttributes transparency = new TransparencyAttributes();
69. transparency.setTransparencyMode(TransparencyAttributes.BLENDED);
70. transparency.setTransparency(0.5f); *// 设置透明度，范围从 0.0（完全不透明）到 1.0（完全透明）*
71. app2.setTransparencyAttributes(transparency);
72. *// 其他属性设置*
73. Material material2 = new Material();
74. material2.setDiffuseColor(new Color3f(0.0f, 1.0f, 0.0f));
75. app2.setMaterial(material2);
76. *//定义总的TransformGroup：transformgroup*
77. TransformGroup transformgroup = new TransformGroup();
78. *//设置对该TransformGroup的读写能力*
79. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
80. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
81. *//将该TransformGroup加入到BranchGroupRoot中*
82. BranchGroupRoot.addChild(transformgroup);
83. *//定义鼠标对场景的旋转、平移与放大功能*
84. *//鼠标旋转功能*
85. MouseRotate mouserotate = new MouseRotate();
86. mouserotate.setTransformGroup(transformgroup);
87. BranchGroupRoot.addChild(mouserotate);
88. mouserotate.setSchedulingBounds(bounds);
89. *//这是之前的不能进行缩放*
90. MouseZoom mousezoom = new MouseZoom();
91. mousezoom.setTransformGroup(transformgroup);
92. BranchGroupRoot.addChild(mousezoom);
93. mousezoom.setSchedulingBounds(bounds);
94. *//重新定义鼠标缩放功能 默认滚轮*
95. MouseWheelZoom mouseWheelZoom = new MouseWheelZoom();
96. mouseWheelZoom.setTransformGroup(transformgroup);
97. BranchGroupRoot.addChild(mouseWheelZoom);
98. mouseWheelZoom.setSchedulingBounds(bounds);
99. *//鼠标平移功能 默认鼠标右键*
100. MouseTranslate mousetranslate = new MouseTranslate();
101. mousetranslate.setTransformGroup(transformgroup);
102. BranchGroupRoot.addChild(mousetranslate);
103. mousetranslate.setSchedulingBounds(bounds);
104. *//        /\*定义一个球体与一个长方体的大小、外观属性与坐标变换，并定义相应的TransformGroup：tg1、tg2\*/*
105. *//        TransformGroup tg1 = new TransformGroup();*
106. *//        tg1.addChild(new Sphere(0.4f, app1));*
107. *//        Transform3D t = new Transform3D();*
108. *//        t.setTranslation(new Vector3f(0.f, -0.425f, 0.f));*
109. *//        TransformGroup tg2 = new TransformGroup(t);*
110. *//        tg2.addChild(new Box(0.5f, 0.005f, 0.5f, app2));*
111. *//*
112. *//        //将定义好的两个TransformGroup(tg1、tg2)加入到总的transformgroup*
113. *//        transformgroup.addChild(tg1);*
114. *//        transformgroup.addChild(tg2);*
115. *//*
116. *//        //对BranchGroupRoot预编译*
117. *//        BranchGroupRoot.compile();*
118. */\*定义一个球体与一个长方体的大小、外观属性与坐标变换，并定义相应的TransformGroup：tg1、tg2\*/*
119. TransformGroup tg1 = new TransformGroup();
120. tg1.addChild(new Sphere(0.4f, app1));
121. *// 创建上方的长方体*
122. TransformGroup tg2Above = new TransformGroup();
123. Transform3D tAbove = new Transform3D();
124. tAbove.setTranslation(new Vector3f(0.f, 0.5f, 0.f)); *// 将长方体放在球的上方*
125. TransformGroup tg2 = new TransformGroup(tAbove);
126. tg2.addChild(new Box(0.5f, 0.005f, 0.5f, app2));
127. *// 创建下方的长方体*
128. TransformGroup tg2Below = new TransformGroup();
129. Transform3D tBelow = new Transform3D();
130. tBelow.setTranslation(new Vector3f(0.f, -0.5f, 0.f)); *// 将长方体放在球的下方*
131. TransformGroup tg3 = new TransformGroup(tBelow);
132. tg3.addChild(new Box(0.5f, 0.005f, 0.5f, app2));
133. *// 创建前方的长方体*
134. TransformGroup tg2Front = new TransformGroup();
135. Transform3D tFront = new Transform3D();
136. tFront.setTranslation(new Vector3f(0.f, 0.f, -0.5f)); *// 将长方体放在球的前方*
137. TransformGroup tg4 = new TransformGroup(tFront);
138. tg4.addChild(new Box(0.5f, 0.5f, 0.005f, app2));
139. *// 创建后方的长方体*
140. TransformGroup tg2Back = new TransformGroup();
141. Transform3D tBack = new Transform3D();
142. tBack.setTranslation(new Vector3f(0.f, 0.f, 0.5f)); *// 将长方体放在球的后方*
143. TransformGroup tg5 = new TransformGroup(tBack);
144. tg5.addChild(new Box(0.5f, 0.5f, 0.005f, app2));
145. *// 创建左边的长方体*
146. TransformGroup tg2Left = new TransformGroup();
147. Transform3D tLeft = new Transform3D();
148. tLeft.setTranslation(new Vector3f(-0.5f, 0.f, 0.f)); *// 将长方体放在球的左边*
149. TransformGroup tg6 = new TransformGroup(tLeft);
150. tg6.addChild(new Box(0.005f, 0.5f, 0.5f, app2));
151. *// 创建右边的长方体*
152. TransformGroup tg2Right = new TransformGroup();
153. Transform3D tRight = new Transform3D();
154. tRight.setTranslation(new Vector3f(0.5f, 0.f, 0.f)); *// 将长方体放在球的右边*
155. TransformGroup tg7 = new TransformGroup(tRight);
156. tg7.addChild(new Box(0.005f, 0.5f, 0.5f, app2));
157. *// 将定义好的所有TransformGroup加入到总的transformgroup*
158. transformgroup.addChild(tg1);
159. transformgroup.addChild(tg2);
160. transformgroup.addChild(tg3);
161. transformgroup.addChild(tg4);
162. transformgroup.addChild(tg5);
163. transformgroup.addChild(tg6);
164. *//        transformgroup.addChild(tg7);*
165. *// 对BranchGroupRoot预编译*
166. BranchGroupRoot.compile();
167. *//通过方法名返回BranchGroupRoot*
168. return BranchGroupRoot;
169. }
170. }

（二）截图：

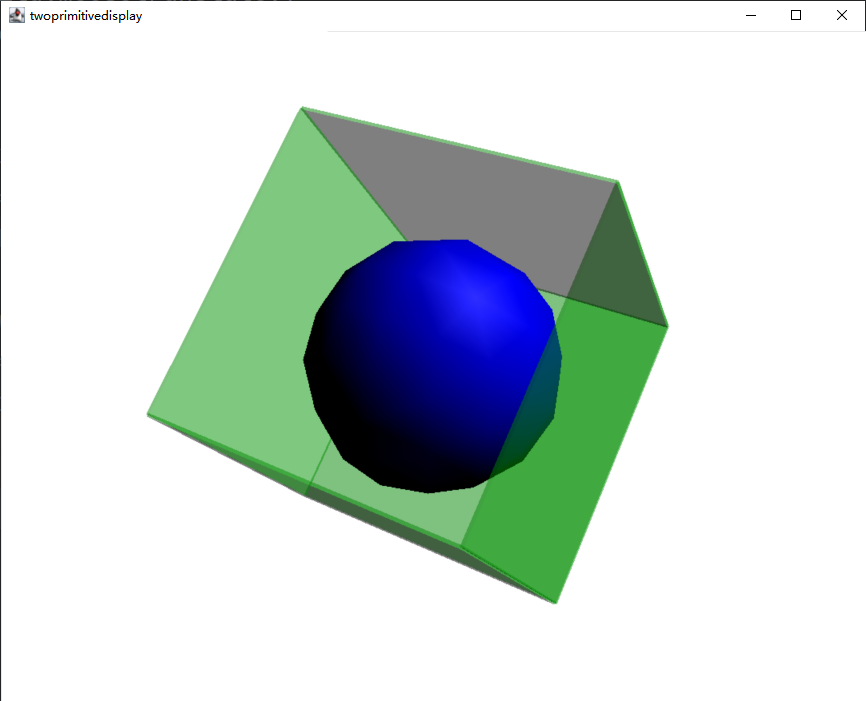


图1- 2 运行结果截图

**三、实验总结**

本次实验是关于使用Java 3D库创建三维图形的实践，主要集中在对球形和长方体组成的组合体进行可视化的编程实现。以下是对本次实验的总结：

1. 实验目标和内容：

本次实验的主要目标是使用Java 3D库创建一个简单的三维场景，其中包括球体和长方体，并添加鼠标交互功能，以便用户可以旋转、平移和缩放场景中的物体。实验涉及了多个关键类，如BranchGroup、TransformGroup、Appearance等。

2. 场景构建：

- 使用BranchGroup作为场景的容器，用于组织和管理三维元素。

- 定义了一个BoundingSphere，用于指定场景的范围。

- 设置背景颜色，为场景提供了背景。

- 添加了平行光源，定义了光源的颜色和照射方向，以影响物体的外观。

3. 三维物体：

- 使用Sphere和Box类创建了一个球体和一个长方体。

- 通过Appearance类定义了物体的外观属性，如颜色。

- 使用TransformGroup类对物体进行了坐标变换，以确定它们的位置和大小。

4. 鼠标交互功能：

- 使用MouseRotate、MouseTranslate和MouseZoom类添加了鼠标交互功能，允许用户通过鼠标操作来改变场景中的视角和物体位置。

1. 界面设置：

- 在twoprimitivedisplay构造函数中设置了显示界面的相关参数，包括布局和Canvas3D的创建。

- 通过SimpleUniverse管理三维场景，并设置默认视图变换。

6. 总结：

通过本次实验，我深入理解了Java 3D编程的基本原理和使用方法。学会了如何创建和配置三维场景，如何定义物体的外观和坐标变换，以及如何添加交互功能以提供用户友好的界面。这个实验为进一步探索三维图形编程和应用提供了坚实的基础。

7. 学习收获：

本次实验让我更加熟悉了Java 3D编程，特别是在构建三维场景和处理交互方面。这对于未来可能涉及到三维可视化的项目或研究非常有用。同时，我也锻炼了代码编写和问题解决的能力。

在未来的学习和实践中，我可以进一步扩展这个基础，创建更复杂的三维场景，并探索更多Java 3D库的功能和特性，以满足不同应用的需求。

**《计算机图形学》实验报告二**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.9.21  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

本实验旨在通过使用Java 3D编程，以一个简单的点的显示程序为例，达到以下目的：

1. 了解并熟练掌握Java 3D编程：通过编写和理解Java 3D程序，学习如何创建和操作三维图形场景。

2. 了解Java 3D基本图形功能：通过创建和显示点集，掌握Java 3D的基本图形绘制功能，包括顶点坐标和点的颜色设置。

3. 了解并熟悉Java 3D的场景式管理：通过创建分支组（BranchGroup）和添加行为（Behavior），学习如何使用Java 3D来管理三维场景，包括旋转、缩放和平移等操作。

1. 通过球体和长方体组合体程序对Java场景式管理实践：虽然代码中没有提到球体和长方体组合体，但可以扩展该程序，以实践更复杂的三维图形对象的创建和管理，例如球体和长方体的组合，从而深入理解Java 3D的场景管理功能。

通过完成这些目标，参与者将获得有关Java 3D编程和三维图形场景管理的基本知识和实践经验。

**二、实验过程**

## 实验内容：

（1）代码

1. */\*\**
2. \* \\* Created with IntelliJ IDEA.
3. \* \\* @ProjectName: 例3.2 点的显示程序实例
4. \* \\* @FileName: DisplayPoints
5. \* \\* @author: li-jihong
6. \* \\* Date: 2023-09-21 13:57
7. \*/
8. import com.sun.j3d.utils.applet.MainFrame;
9. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
10. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
11. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
12. import com.sun.j3d.utils.universe.SimpleUniverse;
13. import javax.media.j3d.\*;
14. import javax.vecmath.Color3f;
15. import javax.vecmath.Point3d;
16. import java.applet.Applet;
17. import java.awt.\*;
18. public class DisplayPoints extends Applet {
19. public DisplayPoints() {
20. setLayout(new BorderLayout());
21. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
22. Canvas3D c = new Canvas3D(gc);
23. add("Center", c);
24. BranchGroup BranchGroupScene = createBranchGroup();
25. SimpleUniverse u = new SimpleUniverse(c);
26. u.getViewingPlatform().setNominalViewingTransform();
27. u.addBranchGraph(BranchGroupScene);
28. }
29. public static void main(String[] args) {
30. new MainFrame(new DisplayPoints(), 450, 450);
31. }
32. public BranchGroup createBranchGroup() {
33. BranchGroup BranchGroupRoot = new BranchGroup();
34. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
35. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
36. Background bg = new Background(bgColor);
37. bg.setApplicationBounds(bounds);
38. BranchGroupRoot.addChild(bg);
39. TransformGroup transformgroup = new TransformGroup();
40. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
41. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
42. BranchGroupRoot.addChild(transformgroup);
43. MouseRotate mouserotate = new MouseRotate();
44. mouserotate.setTransformGroup(transformgroup);
45. BranchGroupRoot.addChild(mouserotate);
46. mouserotate.setSchedulingBounds(bounds);
47. MouseZoom mousezoom = new MouseZoom();
48. mousezoom.setTransformGroup(transformgroup);
49. BranchGroupRoot.addChild(mousezoom);
50. mousezoom.setSchedulingBounds(bounds);
51. MouseTranslate mousetranslate = new MouseTranslate();
52. mousetranslate.setTransformGroup(transformgroup);
53. BranchGroupRoot.addChild(mousetranslate);
54. mousetranslate.setSchedulingBounds(bounds);
55. Shape3D shapepoints = new Shape3D();
56. *//定义顶点坐标*
57. float vertexes[] = {.5f, 0.6f, 0.0f, -0.5f, 0.6f, 0.0f,
58. 0.5f, 0.05f, 0.0f, -0.5f, 0.05f, 0.f,
59. -0.5f, -0.7f, 0.0f, 0.5f, -0.7f, 0.1f,};
60. *//定义点颜色*
61. float pointcolors[] = {1.0f, 0.f, 0.0f, 0.0f, 1.f, 0.0f,
62. 0.0f, 0.f, 1.0f, 1.0f, 1.0f, 0.f,
63. 0.0f, 1.0f, 1.f, 1.f, 0.f, 1.0f};
64. int vCount = 6;
65. PointArray points = new PointArray(vCount, PointArray.COORDINATES |
66. PointArray.COLOR\_3);
67. points.setCoordinates(0, vertexes);
68. points.setColors(0, pointcolors);
69. Appearance app = new Appearance();
70. *//定义点的属性*
71. PointAttributes pointsattributes = new PointAttributes();
72. *//定义点的大小*
73. pointsattributes.setPointSize(70.0f);
74. *//如有下面这项，空间点显示为圆球形；如无，空间点显示为正方形*
75. pointsattributes.setPointAntialiasingEnable(false);
76. app.setPointAttributes(pointsattributes);
77. shapepoints.setGeometry(points);
78. shapepoints.setAppearance(app);
79. transformgroup.addChild(shapepoints);
80. BranchGroupRoot.compile();
81. return BranchGroupRoot;
82. }
83. }
84. 结果截图

|  |  |
| --- | --- |
|  |  |
| 图2-1 使用反走样参数 | 图2-2 未使用反走样参数 |

**三、实验总结**

在本次实验中，我使用了Java 3D编程技术创建了一个简单的三维点的显示程序。这个实验的关键焦点是观察点反走样参数`setPointAntialiasingEnable`的开启和关闭对点的显示形状造成的影响。

1. 实验目的：本实验的主要目标是研究和理解Java 3D编程技术，以及探讨点抗锯齿效果参数在点的显示中的作用。

2. 实验步骤：在实验中，我按照以下步骤进行了操作：

- 首先，我创建了一个Java 3D程序，设置了三维图形的场景，包括窗口布局和视图配置。

- 然后，我创建了一个包含点集的三维场景，其中点的坐标和颜色分别由`vertexes`和`pointcolors`数组定义。

- 接着，我使用了`PointAttributes`来定义了点的属性，其中包括点的大小。

- 最重要的是，我进行了两次实验观察点的显示效果。一次是开启了点反走样效果，另一次是关闭了点反走样效果。

3. 结果比较：在观察实验结果时，我注意到`setPointAntialiasingEnable`参数的状态对点的显示形状产生了显著影响。当该参数开启时，点呈现为圆球形状，边缘更加圆滑；而当关闭该参数时，点呈现为正方形状，边缘更加锐利。这是因为点反走样效果的开启可以平滑点的边缘，使其看起来更加光滑。

4. 实验总结：通过这个实验，我不仅学会了如何使用Java 3D创建三维点集并控制其外观，还深入了解了点反走样效果参数在三维图形显示中的重要作用。这个实验为我提供了有关Java 3D编程技术的基础知识，并突出了反走样效果参数的重要性。根据实际需求，我可以选择开启或关闭反走样效果，以获得不同的视觉效果，从而更好地满足项目的要求。

**《计算机图形学》实验报告三**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.9.28  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

**实验一目的：**

通过编写Java 3D程序，展示如何创建和显示彩色线条，以及如何使用鼠标进行交互操作，包括旋转、平移和缩放。这个程序创建了一个简单的Java 3D场景，其中包含四条彩色线条，并允许用户使用鼠标进行互动。以下是实验的主要目的：

1. 创建一个Java 3D应用程序，用于显示三维图形。

2. 在Java 3D场景中创建一个带有彩色线条的3D对象。

3. 定义线条的顶点坐标和颜色，以在场景中呈现彩色线条。

4. 配置线条的属性，如线宽度和线型。

5. 添加交互性，使用户能够使用鼠标旋转、平移和缩放整个场景。

6. 显示背景颜色，以提高场景的可视效果。

7. 编写一个主类，通过MainFrame启动Java 3D应用程序。

**实验二目的：**

本实验旨在演示如何使用Java 3D库中的LineStripArray类创建和显示带有不同分段的彩色线条，并且通过鼠标交互实现视图的旋转、平移和缩放。以下是实验的主要目的：

1. Java 3D应用程序：创建一个基于Java 3D库的应用程序，用于显示和操作三维图形。

2. 线条创建：使用LineStripArray类创建包含彩色线条的3D对象。本实验中的线条由多个顶点组成，每个顶点都具有坐标和颜色。

3. 线条分段：演示了不同的线条分段方案，其中线条分为不同的子组，以便展示不同的效果。

- 方案一：4个顶点、4个顶点

- 方案二：2个顶点、6个顶点

- 方案三：6个顶点、2个顶点

- 方案四：整个线条作为一个组

4. 线条属性：配置线条的属性，包括线宽度、线型和线条的抗锯齿特性。

5. 鼠标交互：通过MouseRotate、MouseZoom和MouseTranslate等鼠标行为，使用户能够与场景进行交互，改变视图的角度和位置。

6. 背景颜色：设置场景的背景颜色，以提高可视效果。

7. 主类：编写一个主类，通过MainFrame启动Java 3D应用程序，使其可以在窗口中运行。

通过这个实验，可以学到如何使用Java 3D库中的LineStripArray类创建具有不同分段的彩色线条，并实现鼠标交互，以便更好地理解和掌握Java 3D的基本概念和功能。这个实验还可以作为进一步学习和开发更复杂Java 3D应用程序的基础。

**二、实验过程**

## 实验内容一：

1. 代码
2. */\*\**
3. \* \\* Created with IntelliJ IDEA.
4. \* \\* @ProjectName: 例3.4 线显示的程序
5. \* \\* @FileName: DisplayLines
6. \* \\* @author: li-jihong
7. \* \\* Date: 2023-09-28 10:41
8. \*/
9. import com.sun.j3d.utils.applet.MainFrame;
10. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
11. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
12. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
13. import com.sun.j3d.utils.universe.SimpleUniverse;
14. import javax.media.j3d.\*;
15. import javax.vecmath.Color3f;
16. import javax.vecmath.Point3d;
17. import javax.vecmath.Vector3f;
18. import java.applet.Applet;
19. import java.awt.\*;
20. public class DisplayLines extends Applet {
21. public DisplayLines() {
22. setLayout(new BorderLayout());
23. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
24. Canvas3D c = new Canvas3D(gc);
25. add("Center", c);
26. BranchGroup BranchGroupScene = createBranchGroupSceneGraph();
27. SimpleUniverse u = new SimpleUniverse(c);
28. u.getViewingPlatform().setNominalViewingTransform();
29. u.addBranchGraph(BranchGroupScene);
30. }
31. public static void main(String[] args) {
32. new MainFrame(new DisplayLines(), 450, 450);
33. }
34. public BranchGroup createBranchGroupSceneGraph() {
35. BranchGroup BranchGroupRoot = new BranchGroup();
36. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
37. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
38. Background bg = new Background(bgColor);
39. bg.setApplicationBounds(bounds);
40. BranchGroupRoot.addChild(bg);
41. Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);
42. Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);
43. DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);
44. directionalLight.setInfluencingBounds(bounds);
45. BranchGroupRoot.addChild(directionalLight);
46. TransformGroup transformgroup = new TransformGroup();
47. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
48. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
49. BranchGroupRoot.addChild(transformgroup);
50. MouseRotate mouserotate = new MouseRotate();
51. mouserotate.setTransformGroup(transformgroup);
52. BranchGroupRoot.addChild(mouserotate);
53. mouserotate.setSchedulingBounds(bounds);
54. MouseZoom mousezoom = new MouseZoom();
55. mousezoom.setTransformGroup(transformgroup);
56. BranchGroupRoot.addChild(mousezoom);
57. mousezoom.setSchedulingBounds(bounds);
58. MouseTranslate mousetranslate = new MouseTranslate();
59. mousetranslate.setTransformGroup(transformgroup);
60. BranchGroupRoot.addChild(mousetranslate);
61. mousetranslate.setSchedulingBounds(bounds);
62. transformgroup.addChild(shapelines());*//将线加入到TransformGroup中*
63. BranchGroupRoot.compile();
64. return BranchGroupRoot;
65. }
66. public Shape3D shapelines() {
67. Shape3D shapelines0 = new Shape3D();
68. *//定义点坐标*
69. float vertexes[] = {.8f, 0.5f, 0.0f, -0.8f, .8f, 0.0f,
70. 0.8f, 0.0f, 0.0f, -0.8f, 0.5f, 0.0f,
71. 0.8f, -0.5f, 0.0f, -0.8f, 0.2f, 0.0f};
72. *//定义颜色*
73. float colors[] = {1.0f, 0.f, .0f, 0.0f, 1.f, .0f,
74. 0.0f, 0.f, 1.f, 1.0f, 1.0f, 0.f,
75. 0.0f, 1.0f, 1.f, 1.f, 0.f, 1.0f};
76. *//定义线数组*
77. LineArray lines = new LineArray(6, LineArray.COORDINATES | LineArray.COLOR\_3);
78. lines.setCoordinates(0, vertexes);
79. lines.setColors(0, colors);
80. *//定义线属性*
81. LineAttributes lineattributes = new LineAttributes();
82. lineattributes.setLineWidth(20.f);
83. lineattributes.setLineAntialiasingEnable(false);
84. lineattributes.setLinePattern(LineAttributes.PATTERN\_SOLID);
85. *//PATTERN\_SOLID PATTERN\_DASH PATTERN\_DOT PATTERN\_DASH\_DOT*
86. Appearance app = new Appearance();
87. app.setLineAttributes(lineattributes);
88. shapelines0.setGeometry(lines);
89. shapelines0.setAppearance(app);
90. return shapelines0;
91. }
92. }

（2）结果截图

|  |  |
| --- | --- |
|  |  |
| 图3-1 使用反走样参数 | 图3-2 未使用反走样参数 |

## 实验内容二：

（1）代码

1. */\*\**
2. \* \\* Created with IntelliJ IDEA.
3. \* \\* @ProjectName: 例3.6 LineStripArray类程序实例
4. \* \\* @FileName: DisplayLineStrip
5. \* \\* @author: li-jihong
6. \* \\* Date: 2023-09-21 14:12
7. \*/
8. import com.sun.j3d.utils.applet.MainFrame;
9. import com.sun.j3d.utils.behaviors.mouse.MouseRotate;
10. import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;
11. import com.sun.j3d.utils.behaviors.mouse.MouseZoom;
12. import com.sun.j3d.utils.universe.SimpleUniverse;
13. import javax.media.j3d.\*;
14. import javax.vecmath.Color3f;
15. import javax.vecmath.Point3d;
16. import javax.vecmath.Vector3f;
17. import java.applet.Applet;
18. import java.awt.\*;
19. public class DisplayLineStrip extends Applet {
20. public DisplayLineStrip() {
21. setLayout(new BorderLayout());
22. GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();
23. Canvas3D c = new Canvas3D(gc);
24. add("Center", c);
25. BranchGroup BranchGroupScene = createBranchGroupSceneGraph();
26. SimpleUniverse u = new SimpleUniverse(c);
27. u.getViewingPlatform().setNominalViewingTransform();
28. u.addBranchGraph(BranchGroupScene);
29. }
30. public static void main(String[] args) {
31. new MainFrame(new DisplayLineStrip(), 450, 450);
32. }
33. public BranchGroup createBranchGroupSceneGraph() {
34. BranchGroup BranchGroupRoot = new BranchGroup();
35. BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);
36. Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);
37. Background bg = new Background(bgColor);
38. bg.setApplicationBounds(bounds);
39. BranchGroupRoot.addChild(bg);
40. Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);
41. Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);
42. DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);
43. directionalLight.setInfluencingBounds(bounds);
44. BranchGroupRoot.addChild(directionalLight);
45. TransformGroup transformgroup = new TransformGroup();
46. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);
47. transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);
48. BranchGroupRoot.addChild(transformgroup);
49. MouseRotate mouserotate = new MouseRotate();
50. mouserotate.setTransformGroup(transformgroup);
51. BranchGroupRoot.addChild(mouserotate);
52. mouserotate.setSchedulingBounds(bounds);
53. MouseZoom mousezoom = new MouseZoom();
54. mousezoom.setTransformGroup(transformgroup);
55. BranchGroupRoot.addChild(mousezoom);
56. mousezoom.setSchedulingBounds(bounds);
57. MouseTranslate mousetranslate = new MouseTranslate();
58. mousetranslate.setTransformGroup(transformgroup);
59. BranchGroupRoot.addChild(mousetranslate);
60. mousetranslate.setSchedulingBounds(bounds);
61. transformgroup.addChild(Striplines());
62. BranchGroupRoot.compile();
63. return BranchGroupRoot;
64. }
65. public Shape3D Striplines() {
66. Shape3D Striplines0 = new Shape3D();
67. float vertexes[] = {-0.8f, 0.5f, 0.0f, 0.8f, 0.9f, 0.0f,
68. -0.8f, 0.2f, 0.0f, 0.8f, 0.7f, 0.0f,
69. -0.8f, -0.2f, 0.0f, 0.8f, -0.5f, 0.0f,
70. -0.8f, -0.5f, 0.0f, 0.8f, -.8f, 0.0f,};
71. float colors[] = {1.0f, 0.f, .0f, 0.0f, 1.f, .0f,
72. 0.0f, 0.f, 1.f, 1.0f, 1.0f, 0.f,
73. 0.0f, 1.0f, 1.f, 1.f, 0.f, 1.0f,
74. 0.0f, .0f, 0.f, 0.3f, 0.8f, 0.0f,};
75. *//方案一: 分为两组，4个顶点、4个顶点*
76. *//        int[] substrips = new int[2];*
77. *//        substrips[0] = 4;*
78. *//        substrips[1] = 4;*
79. *//方案二: 分为两组，2个顶点、6个顶点*
80. *//        int[] substrips = new int[2];*
81. *//        substrips[0] = 2;*
82. *//        substrips[1] = 6;*
83. *//方案三: 分为两组，6个顶点、2个顶点*
84. *//        int[] substrips = new int[2];*
85. *//        substrips[0] = 6;*
86. *//        substrips[1] = 2;*
87. *//方案四: 整个是一个组*
88. int[] substrips=new int[1];
89. substrips[0]=8;
90. LineStripArray Striplines = new LineStripArray
91. (8, LineArray.COORDINATES | LineArray.COLOR\_3, substrips);
92. Striplines.setCoordinates(0, vertexes);
93. Striplines.setColors(0, colors);
94. LineAttributes lineattributes = new LineAttributes();
95. lineattributes.setLineWidth(9.0f);
96. lineattributes.setLineAntialiasingEnable(true);
97. lineattributes.setLinePattern(0);
98. Appearance app = new Appearance();
99. app.setLineAttributes(lineattributes);
100. Striplines0.setGeometry(Striplines);
101. Striplines0.setAppearance(app);
102. return Striplines0;
103. }
104. }

（2）结果截图

|  |  |
| --- | --- |
|  |  |
| 图3-3分为两组，4个顶点、4个顶点 | 图3-4分为两组，2个顶点、6个顶点 |
|  |  |
| 图3-5分为两组，6个顶点、2个顶点 | 图3-6 整个是一个组 |

**三、实验总结**

**实验一总结：**

在这个实验中，成功地创建了一个简单的Java 3D应用程序，展示了如何创建和显示彩色线条，并实现了鼠标交互操作。以下是实验的主要总结点：

1. Java 3D应用程序：创建了一个基于Java 3D库的应用程序，用于呈现三维图形。

2. 线条的创建：通过定义顶点坐标和颜色，创建了四条彩色线条，并将它们添加到场景中。

3. 线条属性：配置了线条的属性，包括线宽度和线型，以改变线条的外观。

4. 鼠标交互：通过使用Java 3D的鼠标旋转、平移和缩放行为，使用户能够与场景进行交互，改变视图的角度和位置。

5. 背景颜色：设置了场景的背景颜色，以增强可视效果。

6. 主类：编写了一个主类，通过MainFrame启动Java 3D应用程序，使其可以在窗口中运行。

通过这个实验，我学到了如何使用Java 3D库创建基本的三维图形，以及如何实现用户与场景的交互。这是一个入门级的示例，可以作为学习和探索更复杂Java 3D应用程序开发的基础。在实践中，可以进一步扩展这个应用程序，添加更多的三维对象和交互功能，以满足特定需求。

**实验二总结：**

在这个实验中，成功地创建了一个Java 3D应用程序，展示了如何使用LineStripArray类创建和显示带有不同分段的彩色线条，并实现了鼠标交互操作。以下是实验的主要总结点：

1. Java 3D应用程序：创建了一个基于Java 3D库的应用程序，用于呈现三维图形。

2. 线条的创建：通过使用LineStripArray类，创建了一组彩色线条，每个线条由多个顶点组成。还演示了不同的线条分段方案，以展示不同的效果。

3. 线条属性：配置了线条的属性，包括线宽度、线型和抗锯齿特性，以改变线条的外观。

4. 鼠标交互：通过使用Java 3D的鼠标旋转、平移和缩放行为，使用户能够与场景进行交互，改变视图的角度和位置。

5. 背景颜色：设置了场景的背景颜色，以增强可视效果。

6. 主类：编写了一个主类，通过MainFrame启动Java 3D应用程序，使其可以在窗口中运行。

通过这个实验，我学到了如何使用Java 3D库创建具有不同分段的彩色线条，以及如何实现用户与场景的交互。这是一个入门级的示例，可作为学习和探索更复杂Java 3D应用程序开发的基础。进一步的学习可以包括创建更复杂的三维对象，添加纹理和光照效果，以及实现更高级的交互功能，以满足特定需求。通过不断练习和扩展这些概念，可以更深入地理解和应用Java 3D技术。

**《计算机图形学》实验报告四**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.10.12  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

**实验一目的：**

1.学习如何使用Java 3D库创建三维图形应用程序。

2.理解如何创建和配置3D场景，包括添加背景、光照和交互性。

3.熟悉三维图形中的基本概念，如顶点、颜色、光照和多边形渲染。

4.实践创建和渲染基本的三维几何形状，如三角形。

**实验二目的：**

学习如何使用Java 3D库创建三维图形应用程序。

理解如何创建和配置3D场景，包括添加背景、光照和交互性。

熟悉三维图形中的基本概念，如顶点、颜色、光照和多边形渲染。

实践创建和渲染基本的三维几何形状，如三角带。

**实验三目的：**

学习如何使用Java 3D库创建三维图形应用程序。

理解如何创建和配置3D场景，包括添加背景、光照和交互性。

熟悉三维图形中的基本概念，如顶点、颜色、光照和多边形渲染。

实践创建和渲染基本的三维几何形状，如三角扇。

**二、实验过程**

## 实验内容一：

（1）代码

/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: 例3.7 TriangleArray类程序实例  
 \* \\* @FileName: DisplayTriangles  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-09-21 14:14  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class DisplayTriangles extends Applet {  
 public DisplayTriangles() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroup();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new DisplayTriangles(), 450, 450);  
 }  
  
 public BranchGroup createBranchGroup() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new TriangleArrays());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class TriangleArrays extends Shape3D {  
 public TriangleArrays() {  
 int vCount = 12;  
 float vertexes[] = {-0.6f, 0.9f, 0.0f, -0.6f, -0.9f, 0.2f,  
 -0.4f, 0.9f, -0.2f, -0.2f, -0.9f, 0.2f,  
 0.0f, 0.9f, -0.2f, 0.0f, -0.9f, 0.2f,  
 0.2f, 0.7f, 0.0f, 0.2f, -0.9f, 0.3f,  
 0.5f, 0.8f, -0.3f, 0.6f, -.9f, 0.0f,  
 0.8f, 0.9f, 0.2f, 0.8f, -0.8f, 0.3f};  
 float colors[] = {0.0f, 0.5f, 1.0f, 0.0f, 0.5f, 1.0f,  
 0.0f, 0.8f, .0f, 1.0f, 0.0f, 0.3f,  
 0.0f, 1.0f, 0.5f, 0.9f, 1.0f, 0.0f,  
 0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.5f,  
 1.0f, 0.8f, 0.0f, 1.0f, 0.5f, 0.0f};  
 TriangleArray trianglearray = new TriangleArray  
 (vCount, TriangleArray.COORDINATES | TriangleArray.COLOR\_3);  
 trianglearray.setCoordinates(0, vertexes);  
 trianglearray.setColors(0, colors);  
 PolygonAttributes polygonattributes = new PolygonAttributes();  
// polygonattributes.setCullFace(PolygonAttributes.CULL\_NONE);  
// polygonattributes.setCullFace(PolygonAttributes.CULL\_FRONT);  
 polygonattributes.setCullFace(PolygonAttributes.CULL\_BACK);  
 Appearance app = new Appearance();  
 app.setPolygonAttributes(polygonattributes);  
 this.setGeometry(trianglearray);  
 this.setAppearance(app);  
 }  
}

（2）结果截图

|  |  |  |
| --- | --- | --- |
|  |  |  |
| CULL\_NONE | CULL\_FRONT | CULL\_BACK |

## 实验内容二：

（1）代码

package week7\_fourth.\_3\_8; /\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: 例3.8 TriangleStripArray类程序实例  
 \* \\* @FileName: week7\_fourth.\_3\_8.TriangleStripArrays  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-09-21 14:15  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class TriangleStripArrays extends Applet {  
 public TriangleStripArrays() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroup();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new TriangleStripArrays(), 450, 450);  
 }  
  
 public BranchGroup createBranchGroup() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new TriangleStrip());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class TriangleStrip extends Shape3D {  
 public TriangleStrip() {  
 int vertexesCount = 12;  
 int stripCount[] = new int[1];  
 float vertexes[] = {-0.9f, 0.9f, 0.0f, -0.8f, -0.9f, 0.2f,  
 -0.6f, 0.8f, -0.2f, -0.4f, -0.8f, 0.2f,  
 -0.3f, 0.9f, -0.2f, -0.2f, -0.9f, 0.2f,  
 0.4f, 0.7f, 0.0f, 0.4f, -0.7f, 0.3f,  
 0.6f, 0.9f, -0.3f, 0.6f, -.9f, 0.0f,  
 0.9f, 0.8f, 0.2f, 0.8f, -0.8f, 0.3f};  
 float colors[] = {0.0f, 0.5f, 1.0f, 0.0f, 0.5f, 1.0f,  
 0.0f, 0.8f, .0f, 1.0f, 0.0f, 0.3f,  
 0.0f, 1.0f, 0.5f, 0.9f, 1.0f, 0.0f,  
 0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.5f,  
 1.0f, 0.8f, 0.0f, 1.0f, 0.5f, 0.0f};  
 //生成三个子Strip，每个含有四个顶点  
 stripCount[0] = 12;  
// stripCount[1] = 6;  
// stripCount[2] = 3;  
 TriangleStripArray triangleStriparray = new TriangleStripArray(vertexesCount,  
 TriangleStripArray.COORDINATES | TriangleStripArray.COLOR\_3, stripCount);  
 triangleStriparray.setCoordinates(0, vertexes);  
 triangleStriparray.setColors(0, colors);  
 PolygonAttributes polygonattributes = new PolygonAttributes();  
 polygonattributes.setCullFace(PolygonAttributes.CULL\_NONE);  
 Appearance app = new Appearance();  
 app.setPolygonAttributes(polygonattributes);  
 this.setGeometry(triangleStriparray);  
 this.setAppearance(app);  
 }  
}

（2）结果截图

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 三个Strip，每个含有顶点数目（4,4,4） | 三个Strip，每个含有顶点数目（3,6,3） | 三个Strip，每个含有顶点数目（3,4,5） |  |
|  |  |  |  |
| 两个Strip，每个含有顶点数目（3,9） | 三个Strip，每个含有顶点数目（4,8） | 三个Strip，每个含有顶点数目（5,7） | 三个Strip，每个含有顶点数目（6,6） |
|  | | | |
| 一个Strip，每个含有顶点数目（12） | | | |

## 实验内容三：

（1）代码

package week7\_fourth.\_3\_9;  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.\*;  
import java.applet.Applet;  
import java.awt.\*;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: 例3.9 TriangleFanArray类程序实例  
 \* \\* @FileName: week7\_fourth.\_3\_9.TriangleFanArray  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-10-12 20:11  
 \*/  
  
public class TriangleFanArray2 extends Applet {  
 public TriangleFanArray2() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroup();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new TriangleFanArray2(), 450, 450);  
 }  
  
 public BranchGroup createBranchGroup() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new ShapeTriangleFanArray());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class ShapeTriangleFanArray extends Shape3D {  
 public ShapeTriangleFanArray() {  
 int vertexesCount = 12;  
// int stripCount[] = new int[1];  
// int stripCount[]=new int[2];  
 int stripCount[]=new int[3];  
 float vertexes[] = {.0f, 0.9f, 0.0f, -1.f, -0.8f, 0.f,  
 -0.8f, -0.6f, -0.2f, -0.6f, -0.9f, 0.2f,  
 -0.4f, -0.8f, -0.2f, 0.f, -0.8f, 0.2f,  
 0.2f, -0.5f, 0.0f, 0.4f, -0.6f, -0.5f,  
 0.6f, -0.8f, -0.3f, 0.8f, -0.9f, -0.2f,  
 0.9f, -0.7f, -0.2f, 1.1f, -0.8f, -0.3f};  
 float colors[] = {0.0f, 0.5f, 1.0f, 0.0f, 0.5f, 1.0f,  
 0.0f, 0.8f, .0f, 1.0f, 0.0f, 0.3f,  
 0.0f, 1.0f, 0.5f, 0.9f, 1.0f, 0.0f,  
 0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.5f,  
 1.0f, 0.8f, 0.0f, 1.0f, 0.5f, 0.0f};  
 stripCount[0] = 4;  
 stripCount[1] = 4;  
 stripCount[2] = 4;  
 TriangleFanArray triangleFanarray = new TriangleFanArray(vertexesCount,  
 TriangleFanArray.COORDINATES | TriangleFanArray.COLOR\_3, stripCount);  
 triangleFanarray.setCoordinates(0, vertexes);  
 triangleFanarray.setColors(0, colors);  
 PolygonAttributes polygonattributes = new PolygonAttributes();  
 polygonattributes.setCullFace(PolygonAttributes.CULL\_NONE);  
  
// polygonattributes.setPolygonMode(PolygonAttributes.POLYGON\_LINE); // 设置为线模型  
  
 Appearance app = new Appearance();  
 app.setPolygonAttributes(polygonattributes);  
 this.setGeometry(triangleFanarray);  
 this.setAppearance(app);  
 }  
}

（2）结果截图

|  |  |
| --- | --- |
|  |  |
| 一个Fan | 与其线模型 |
|  |  |
| 两个Fan | 三个Fan |
|  |  |
| （4,8）Fan | （4,4,4）Fan |

**三、实验总结**

**实验一总结：**

在这个实验中，创建了一个使用Java 3D库的三维图形应用程序，名为"DisplayTriangles"。以下是实验的总结：

创建三维场景：使用Java 3D库创建了一个三维场景。这包括设置背景颜色和光照条件，以提供更真实的外观。

绘制三角形：通过创建TriangleArrays类，绘制了一组三角形。每个三角形由一系列顶点和颜色组成。这些三角形组成了我们的三维几何形状。

添加交互性：使用Java 3D库的鼠标交互行为类（MouseRotate、MouseZoom和MouseTranslate）来实现用户与场景的交互。这允许用户旋转、缩放和平移整个场景，以便更好地观察三维图形。

实验目的达成：通过完成这个实验，我们达到了以下目标：

学习了如何使用Java 3D库创建基本的三维图形应用程序。

理解了三维场景的配置和渲染，包括光照和背景设置。

掌握了三维几何形状的创建和呈现，包括顶点和颜色的设置。

实践了用户与三维场景的交互性实现。

总的来说，这个实验帮助初步掌握了Java 3D库的使用，并为进一步研究和开发复杂的三维图形应用程序打下了基础。

**实验二总结：**

实验总结： 在这个实验中，创建了一个使用Java 3D库的三维图形应用程序，名为"TriangleStripArrays"。以下是实验的总结：

创建三维场景：使用Java 3D库创建了一个三维场景。这包括设置背景颜色和光照条件，以提供更真实的外观。

绘制三角带：通过创建TriangleStrip类，绘制了一个三角带。每个三角带由一系列顶点和颜色组成。这些三角带组成了三维几何形状。

添加交互性：使用Java 3D库的鼠标交互行为类（MouseRotate、MouseZoom和MouseTranslate）来实现用户与场景的交互。这允许用户旋转、缩放和平移整个场景，以便更好地观察三维图形。

实验目的达成：通过完成这个实验，达到了以下目标：

学习了如何使用Java 3D库创建基本的三维图形应用程序。

理解了三维场景的配置和渲染，包括光照和背景设置。

掌握了三维几何形状的创建和呈现，包括顶点和颜色的设置。

实践了用户与三维场景的交互性实现。

总的来说，这个实验帮助学习者初步掌握了Java 3D库的使用，并为进一步研究和开发复杂的三维图形应用程序打下了基础。

**实验三总结：**

当分析你的Java 3D实验时，可以关注一些更详细的方面，包括以下内容：

1. 导入的包：你导入了`com.sun.j3d`和`javax.media.j3d`等包，这些包是Java 3D库的一部分，用于构建和渲染三维图形对象。

2. 构造函数：你的`TriangleFanArray2`类有一个构造函数，用于设置和初始化应用程序的基本组件。这包括创建`Canvas3D`用于渲染，创建`BranchGroup`作为场景图的根等。

3. 主方法：`main`方法用于启动Java 3D应用程序，创建`TriangleFanArray2`实例，并在窗口中显示它。这是Java应用程序的入口点。

4. 创建场景：`createBranchGroup`方法负责创建场景图。这包括设置背景、光源、交互行为（如旋转、平移、缩放）以及`ShapeTriangleFanArray`，它是包含你自定义的三维形状的`Shape3D`对象。

5. 自定义三维形状：`ShapeTriangleFanArray`类的构造函数用于创建三维图形对象。在这里，你定义了顶点坐标、颜色和三角形扇形的顶点连接方式。你还设置了多边形属性，控制了多边形的渲染方式。

6. 线模型：在注释中提到你尝试设置线模型渲染，但它是注释掉的。如果你想要以线框模式渲染，你可以取消注释`polygonattributes.setPolygonMode(PolygonAttributes.POLYGON\_LINE);`这一行，这将使多边形以线条形式显示而不是填充。

7. 编译和显示：最后，你使用`compile`方法编译`BranchGroup`，然后返回它，以便在主程序中添加到Java 3D视图中。你的应用程序使用`SimpleUniverse`来设置视图平台并显示三维场景。

总之，这个实验涵盖了创建、渲染和交互式查看三维图形的基本步骤，包括了许多Java 3D编程的关键方面。

**《计算机图形学》实验报告五**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.10.19  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

**实验一目的：**

这个示例的学习目的是介绍如何使用Java 3D库创建一个简单的三维图形，并学习以下内容：

如何创建Java 3D应用程序。

如何使用IndexedTriangleArray类来定义三维模型的几何和颜色属性。

如何设置背景和光照效果。

如何使用鼠标交互控制旋转、缩放和平移。

**实验二目的：**

如何创建Java 3D应用程序。

如何使用QuadArray类来定义四边面的几何和颜色属性。

如何设置背景和光照效果。

如何使用鼠标交互控制旋转、缩放和平移。

**实验三目的：**

如何创建Java 3D应用程序。

如何添加透明物体并设置透明属性。

如何添加环境光源，改善渲染效果。

如何使用鼠标交互控制旋转、缩放和平移。

**二、实验过程**

## 实验内容一：

（1）代码

package week8\_fifth.\_3\_13;  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: IndexedTriangleArray\_1  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 12:56  
 \*/  
  
public class IndexedTriangleArray\_1 extends Applet {  
 public IndexedTriangleArray\_1() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroup();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new IndexedTriangleArray\_1(), 450, 450);  
 }  
  
 public BranchGroup createBranchGroup() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new ShapeIndexedTriangleArray());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class ShapeIndexedTriangleArray extends Shape3D {  
 public ShapeIndexedTriangleArray() {  
 int vertexesCount = 12;  
 int indexcount = 6;  
 int[] index = {0, 1, 3, 5, 7, 9};  
 float vertexes[] = {-.9f, 0.8f, 0.0f, -0.8f, -0.8f, 0.2f,  
 -0.6f, -0.4f, -0.2f, -0.4f, -0.9f, 0.2f,  
 -0.2f, 0.8f, -0.2f, 0.f, -0.8f, 0.2f,  
 0.2f, -0.5f, 0.0f, 0.4f, 0.6f, -0.5f,  
 0.6f, -0.8f, -0.3f, 0.8f, -0.9f, -0.2f,  
 0.9f, 0.7f, -0.2f, 1.1f, -0.8f, -0.3f};  
 float colors[] = {0.0f, 0.5f, 1.0f, 0.0f, 0.5f, 1.0f,  
 0.0f, 0.8f, .0f, 1.0f, 0.0f, 0.3f,  
 0.0f, 1.0f, 0.5f, 0.9f, 1.0f, 0.0f,  
 0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.5f,  
 1.0f, 0.8f, 0.0f, 1.0f, 0.5f, 0.0f};  
 IndexedTriangleArray indextrianglearray = new IndexedTriangleArray(vertexesCount,  
 IndexedTriangleArray.COORDINATES | IndexedTriangleArray.COLOR\_3, indexcount);  
 indextrianglearray.setCoordinates(0, vertexes);  
 indextrianglearray.setColors(0, colors);  
 indextrianglearray.setCoordinateIndices(0, index);  
 indextrianglearray.setColorIndices(0, index);  
 PolygonAttributes polygonattributes = new PolygonAttributes();  
 polygonattributes.setCullFace(PolygonAttributes.CULL\_NONE);  
 Appearance app = new Appearance();  
 app.setPolygonAttributes(polygonattributes);  
 this.setGeometry(indextrianglearray);  
 this.setAppearance(app);  
 }  
}

（2）结果截图

|  |
| --- |
|  |
| IndexedTriangleArray |

## 实验内容二：

（1）代码

package week8\_fifth;  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: QuadArray\_1  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 13:08  
 \*/  
  
public class QuadArray\_1 extends Applet {  
 public QuadArray\_1() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroup();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new QuadArray\_1(), 450, 450);  
 }  
  
 public BranchGroup createBranchGroup() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new ShapeQuadArray());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class ShapeQuadArray extends Shape3D {  
 public ShapeQuadArray() {  
 int vertexCount = 12;  
 float vertexes[] = {-0.8f, 0.9f, 0.0f, -0.8f, -0.8f, 0.f,  
 -0.6f, -0.8f, 0.f, -0.6f, 0.9f, 0.f,  
 -0.4f, 0.9f, 0.f, -0.4f, -0.7f, -0.9f,  
 0.4f, -0.8f, 0.f, 0.4f, 0.8f, 0.0f,  
 0.5f, 0.8f, 0.f, 0.6f, -0.8f, 0.0f,  
 0.8f, -0.7f, 0.f, 0.8f, 0.8f, 0.f};  
 float colors[] = {0.0f, 0.5f, 1.0f, 0.0f, 0.5f, 1.0f,  
 0.0f, 0.8f, .0f, 1.0f, 0.0f, 0.3f,  
 0.0f, 1.0f, 0.5f, 0.9f, 1.0f, 0.0f,  
 0.5f, 0.0f, 1.0f, 0.0f, 0.5f, 1.0f,  
 1.0f, 0.5f, 0.0f, 1.0f, 0.0f, 0.5f,  
 1.0f, 0.8f, 0.0f, 1.0f, 0.5f, 0.0f};  
 QuadArray quadarray = new QuadArray(vertexCount,  
 QuadArray.COORDINATES | QuadArray.COLOR\_3);  
 quadarray.setCoordinates(0, vertexes);  
 quadarray.setColors(0, colors);  
 PolygonAttributes polygonattributes = new PolygonAttributes();  
 polygonattributes.setCullFace(PolygonAttributes.CULL\_NONE);  
 //polygonattributes.setCullFace(PolygonAttributes.CULL\_BACK);  
 //polygonattributes.setCullFace(PolygonAttributes.CULL\_FRONT);  
 Appearance app = new Appearance();  
 app.setPolygonAttributes(polygonattributes);  
 this.setGeometry(quadarray);  
 this.setAppearance(app);  
 }  
}

（2）结果截图

|  |  |
| --- | --- |
|  |  |

## 实验内容三：

（1）代码

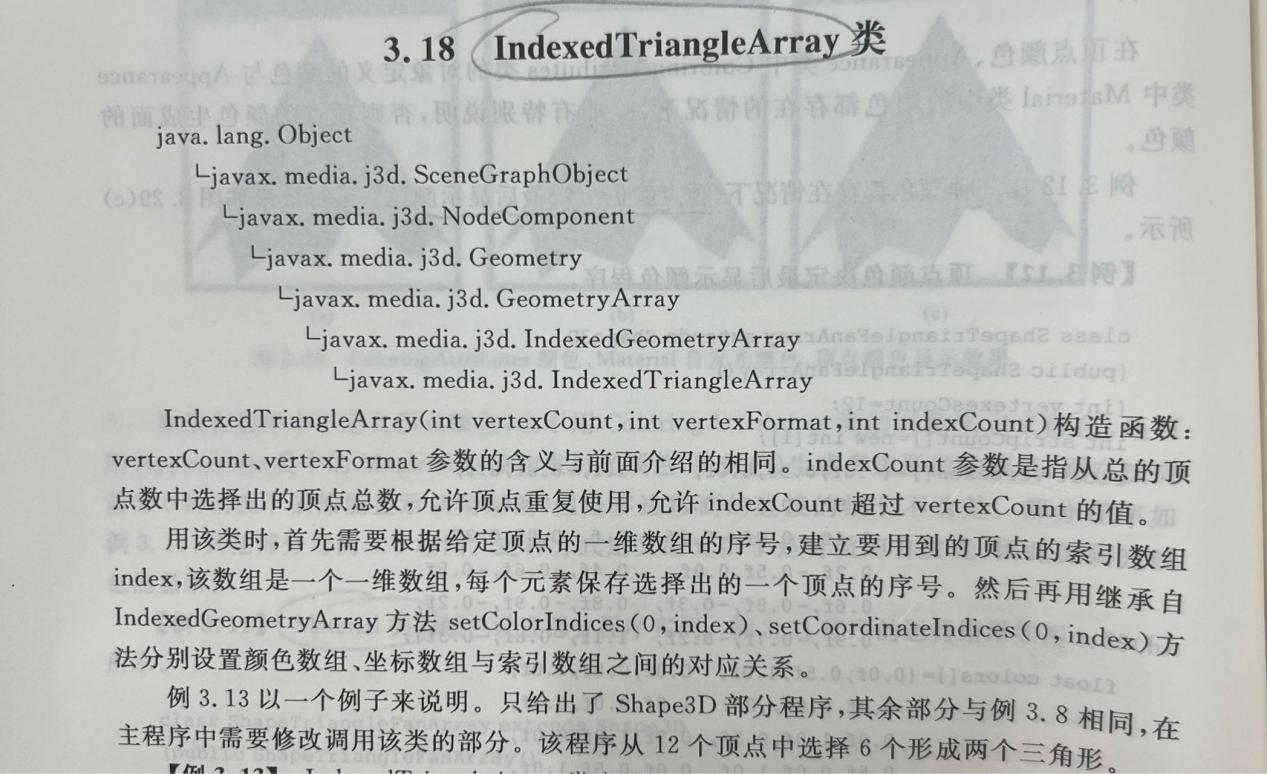
package week8\_fifth.\_3\_16;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: TransparencyAttributess  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 13:13  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.geometry.Box;  
import com.sun.j3d.utils.geometry.Sphere;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class TransparencyAttributess extends Applet {  
 public TransparencyAttributess() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new TransparencyAttributess(), 300, 300);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BrachGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BrachGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 1.f, 1.f);  
 Vector3f vec = new Vector3f(-1.f, -1.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BrachGroupRoot.addChild(directionalLight);  
 Appearance app1 = new Appearance();  
 Material material1 = new Material();  
 material1.setDiffuseColor(new Color3f(1.0f, .0f, 0.0f));  
 app1.setMaterial(material1);  
//定义球体的透明度，透明模式选NICEST=1，透明度0.6f  
 TransparencyAttributes transparence = new TransparencyAttributes(1, .6f);  
 app1.setTransparencyAttributes(transparence);  
 Appearance app2 = new Appearance();  
 Material material2 = new Material();  
 material2.setDiffuseColor(new Color3f(.0f, 1.0f, 0.0f));  
 app2.setMaterial(material2);  
 TransparencyAttributes transparence1 = new TransparencyAttributes(1, .8f);  
 app2.setTransparencyAttributes(transparence1);  
 Appearance app3 = new Appearance();  
 Material material3 = new Material();  
 material3.setDiffuseColor(new Color3f(.0f, .0f, 1.0f));  
 app3.setMaterial(material3);  
 app3.setTransparencyAttributes(transparence1);  
 Appearance app4 = new Appearance();  
 Material material4 = new Material();  
 material4.setDiffuseColor(new Color3f(.0f, 1.0f, 1.0f));  
 app4.setMaterial(material2);  
 app4.setTransparencyAttributes(transparence1);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BrachGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BrachGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BrachGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BrachGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 TransformGroup tg1 = new TransformGroup();  
 tg1.addChild(new Sphere(.9f, 1, 100, app1));  
 Transform3D t = new Transform3D();  
 t.setTranslation(new Vector3f(-0.2f, 0.1f, 0.2f));  
 TransformGroup tg2 = new TransformGroup(t);  
 tg2.addChild(new Box(0.2f, 0.2f, 0.2f, app2));  
 Transform3D t1 = new Transform3D();  
 t1.setTranslation(new Vector3f(-0.2f, 0.1f, 0.2f));  
 TransformGroup tg3 = new TransformGroup(t1);  
 tg3.addChild(new Sphere(.6f, 1, 100, app3));  
 Transform3D t2 = new Transform3D();  
 t2.setTranslation(new Vector3f(0.4f, 0.2f, -0.4f));  
 TransformGroup tg4 = new TransformGroup(t2);  
 tg4.addChild(new Sphere(.3f, 1, 100, app4));  
 transformgroup.addChild(tg1);  
 transformgroup.addChild(tg2);  
 transformgroup.addChild(tg3);  
 transformgroup.addChild(tg4);  
 BrachGroupRoot.compile();  
 return BrachGroupRoot;  
 }  
}

（2）结果截图

|  |  |
| --- | --- |
|  |  |

**三、实验总结**

**实验一总结：**

****

创建Java 3D应用程序：可以学习如何创建一个Java 3D应用程序，包括设置窗口和创建3D场景。

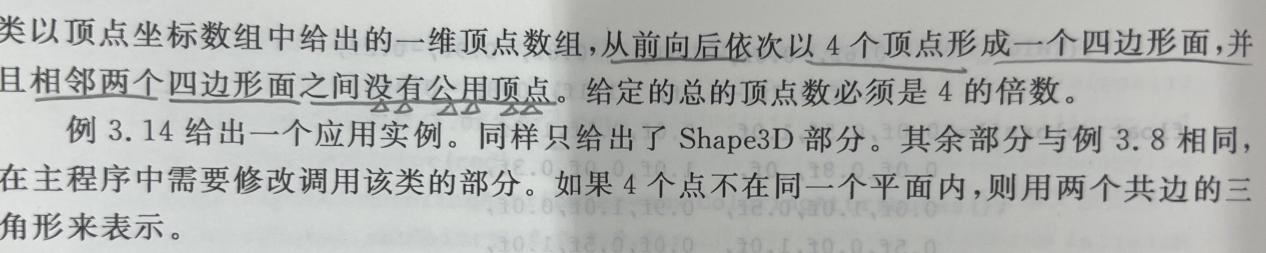
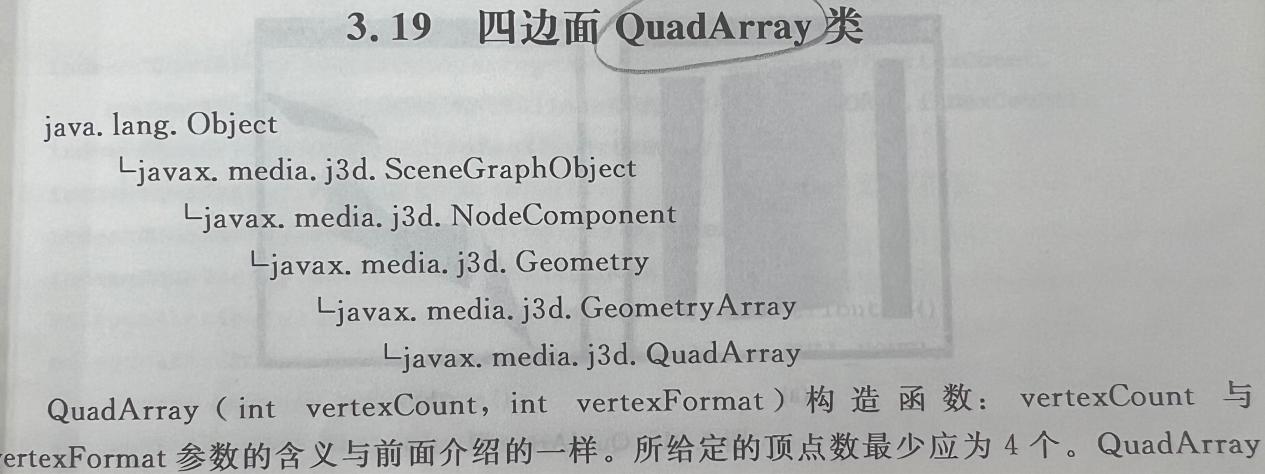
使用IndexedTriangleArray类：了解了如何使用IndexedTriangleArray类来定义三维模型的顶点坐标和颜色信息。这是构建3D模型的基本步骤之一。

背景和光照：学会了如何设置背景颜色以及如何添加定向光源来影响模型的照明效果。这是创建逼真三维场景的关键。

鼠标交互：通过添加鼠标旋转、缩放和平移的交互功能，可以学习如何使用户能够以交互方式浏览和操作三维场景。

渲染和显示：最后，学会了如何将创建的3D模型添加到场景中，并通过设置外观（Appearance）来定义模型的渲染属性。

**实验二总结：**

****

创建Java 3D应用程序：可以学习如何创建一个Java 3D应用程序，包括设置窗口和创建3D场景。

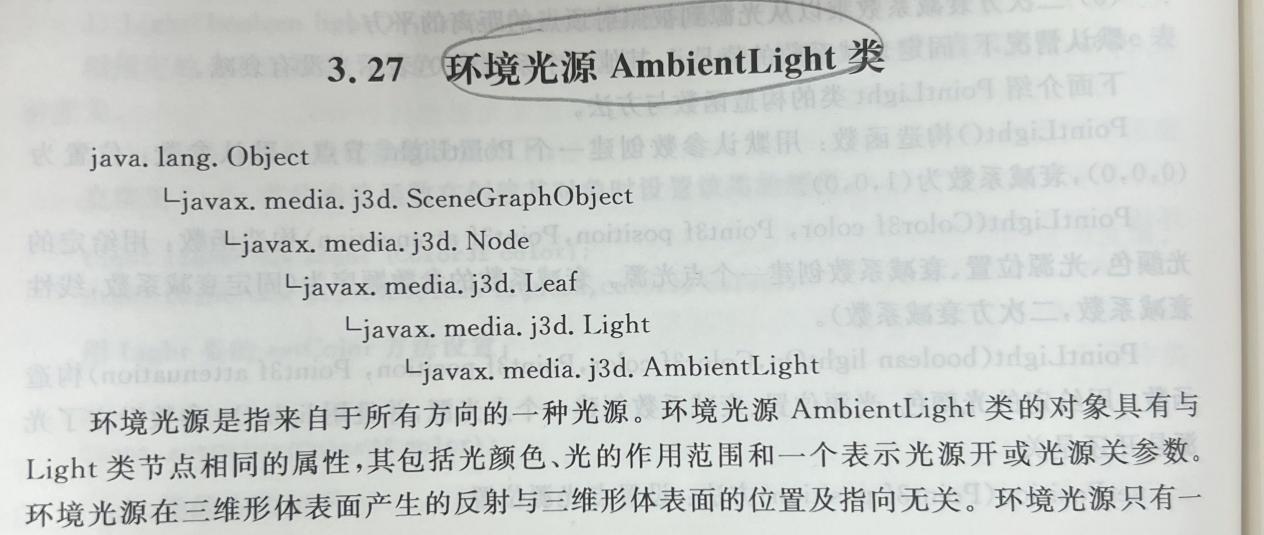
使用QuadArray类：了解了如何使用QuadArray类来定义四边面的顶点坐标和颜色信息。这是构建3D模型的基本步骤之一。

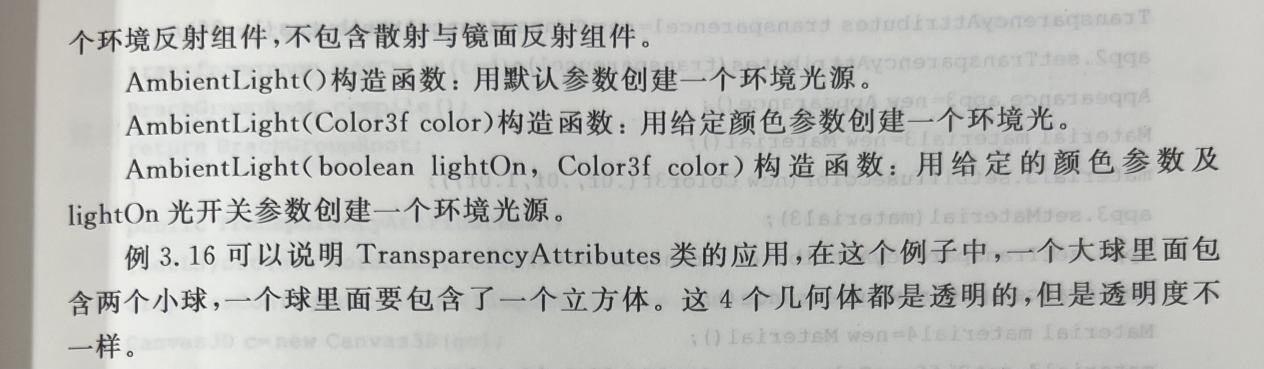
背景和光照：学会了如何设置背景颜色以及如何添加定向光源来影响模型的照明效果。这是创建逼真三维场景的关键。

鼠标交互：通过添加鼠标旋转、缩放和平移的交互功能，可以学习如何使用户能够以交互方式浏览和操作三维场景。

渲染和显示：最后，学会了如何将创建的四边面添加到场景中，并通过设置外观（Appearance）来定义模型的渲染属性。

**实验三总结：**





创建Java 3D应用程序：学习了如何创建一个Java 3D应用程序，包括设置窗口和创建3D场景。

透明物体和透明属性：了解了如何使用QuadArray类创建透明的三维物体，并设置不同的透明度，以实现透明效果。这是创建逼真的透明物体的关键。

环境光源：学会了如何添加环境光源，以改善整个场景的渲染效果。环境光源通常用于减轻阴影和强调物体的轮廓。

鼠标交互：通过添加鼠标旋转、缩放和平移的交互功能，可以学习如何使用户能够以交互方式浏览和操作三维场景。

渲染和显示：最后，学会了如何将创建的透明物体和环境光源添加到场景中，并通过设置外观（Appearance）来定义物体的渲染属性。

**《计算机图形学》实验报告六**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.11.02  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

**实验一目的：**

1.学习如何使用Java 3D库创建和渲染三维图形。

2.掌握如何使用Java 3D库的不同组件，如`Canvas3D`、‘BranchGroup'、

'SimpleUniverse '、"Shape3D等，来构建三维场景。

3.熟悉如何实现鼠标交互功能，如旋转、平移和缩放，以便用户可以与三维场景进行交互。4.理解如何创建几何形状、设置外观（包括材质、颜色等)以及应用光照效果。

**实验二目的：**

1.学习如何使用Java 3D库创建和渲染三维图形，特别是球体的表面。

2.熟悉如何使用不同的几何数组类型(TriangleArray和QuadArray)来表示三维对象的不同部

分。

3.理解如何计算球体表面上的点的坐标，并为这些点设置法向量，以便在渲染时产生光照效

果。

4.学习如何实现鼠标交互功能，如旋转、平移和缩放，以便用户可以与三维对象互动。

**实验三目的：**

创建和展示三阶贝塞尔曲线的三维可视化。它包括了一个简单的三维场景，其中包含了贝塞尔曲线的可视化对象。

**实验四目的：**

实现了一个三阶贝塞尔曲面的可视化，包括控制顶点和曲面网格的绘制。

1. **实验过程**

**理论知识：**

1. **Bezier曲线的定义**

给定空间n+1个点P0、P1、P2…Pn,则可生成一条n次的Bezier曲线。



Bernstein基函数：



其中，P0、P1、P2…Pn称为n次的Bezier曲线的控制顶点；

u称为Bezier曲线的参数；

从公式中可看出，Bezier曲线的次数随着控制顶点个数的增加而增加。其中，最常用的是2次、3次Bezier曲线。通过调整控制顶点的位置，可改变曲线的形状。

**（2）Bernstein基函数的性质**

①非负性

②规范性

③对称性

④端点性

⑤可导性

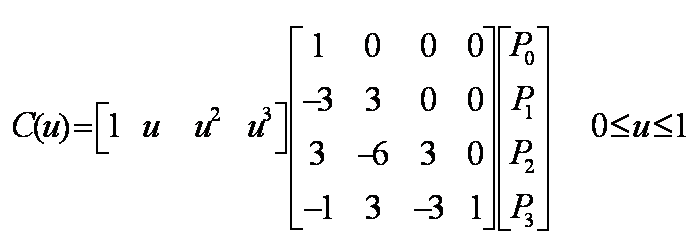
⑥基函数的最值

**（3）三次Bezier曲线的矩阵表示形式**

Bezier曲线的矩阵表示形式的优点就是便于计算机编程实现。如果给定4个控制顶点P0、P1、P2、P3，此时，n=3，则可生成一条三次Bezier曲线。三次Bezier曲线代数表示形式如下：



矩阵表示形式：



## 实验内容一：

（1）代码

package week10\_sixth.\_3\_17;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: RotationQuadarray  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 10:45  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Point3f;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class RotationQuadarray extends Applet {  
 public RotationQuadarray() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String argv[]) {  
 new MainFrame(new RotationQuadarray(), 400, 400);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 Shape3D RotationSurface = new SurfaceDisplay();  
 transformgroup.addChild(RotationSurface);  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class SurfaceDisplay extends Shape3D {  
 float[][][] SurfacePointsxyz = new float[5][51][3];  
  
 SurfaceDisplay() {  
 this.setGeometry(createGeometry0());  
 this.setAppearance(createAppearance0());  
 }  
  
 Geometry createGeometry0() {  
 int i, j, k, c;  
 int n0 = 50;  
 float theta;  
 //定义在xoy平面内的旋转线，旋转轴为y轴  
 float[] Xp0 = {.3f, .4f, .5f, .6f, .9f};  
 float[] Yp0 = {-.4f, -.2f, 0.f, .2f, .5f};  
 //计算对圆周n0等分后所得的旋转角  
 theta = 2.f \* (float) Math.PI / n0;  
 //计算旋转后旋转平面上点的x、y、z坐标值  
 for (i = 0; i < 5; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 SurfacePointsxyz[i][j][0] = Xp0[i] \* (float) Math.cos(theta \* j);  
 SurfacePointsxyz[i][j][1] = Yp0[i];  
 SurfacePointsxyz[i][j][2] = Xp0[i] \* (float) Math.sin(theta \* j);  
 }  
 //按顺时针方向设置每个四边形点的坐标值，求法向量  
 QuadArray SurfaceQuadArray =  
 new QuadArray(5 \* n0 \* 4, GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 c = 0;//该变量用来对顶点按顺序编号，该编号是连续的，不能重复  
 for (i = 0; i < 4; i++) {  
 for (j = 0; j < n0; j++) {  
 Point3f A = new Point3f(SurfacePointsxyz[i][j][0],  
 SurfacePointsxyz[i][j][1], SurfacePointsxyz[i][j][2]);  
 Point3f B = new Point3f(SurfacePointsxyz[i + 1][j][0],  
 SurfacePointsxyz[i + 1][j][1], SurfacePointsxyz[i + 1][j][2]);  
 Point3f C = new Point3f(SurfacePointsxyz[i + 1][j + 1][0],  
 SurfacePointsxyz[i + 1][j + 1][1], SurfacePointsxyz[i + 1][j + 1][2]);  
 Point3f D = new Point3f(SurfacePointsxyz[i][j + 1][0],  
 SurfacePointsxyz[i][j + 1][1], SurfacePointsxyz[i][j + 1][2]);  
 //计算四个点的法向量，使法向量指向体外  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
 //设置点的序号  
 SurfaceQuadArray.setCoordinate(c, A);  
 SurfaceQuadArray.setCoordinate(c + 1, B);  
 SurfaceQuadArray.setCoordinate(c + 2, C);  
 SurfaceQuadArray.setCoordinate(c + 3, D);  
 //设置点的序号所对应的法向量  
 SurfaceQuadArray.setNormal(c, n);  
 SurfaceQuadArray.setNormal(c + 1, n);  
 SurfaceQuadArray.setNormal(c + 2, n);  
 SurfaceQuadArray.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 }  
 return SurfaceQuadArray;  
 }  
  
 Appearance createAppearance0() {//指定外观，这样才有明暗效果  
 PolygonAttributes polygona = new PolygonAttributes();  
 //有了下面这两行语句，在有法向量的情况下，可使面的两侧都能显示  
 polygona.setBackFaceNormalFlip(true);  
 polygona.setCullFace(PolygonAttributes.CULL\_NONE);  
 //polygona.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 //polygona.setPolygonMode(PolygonAttributes.POLYGON\_POINT);  
 Appearance appearance = new Appearance();  
 appearance.setPolygonAttributes(polygona);  
 Material material = new Material();  
 Color3f white = new Color3f(1.0f, .0f, .0f);  
 Color3f red = new Color3f(.0f, .0f, 1.0f);  
 material.setDiffuseColor(white);  
 //material.setSpecularColor(red);  
 //material.setShininess(20.0f);  
 appearance.setMaterial(material);  
 return appearance;  
 }  
}

（2）结果截图（四边网格曲面模型）

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 面模型 | 线模型 | 点模型 |

## 实验内容二：

1. 代码

package week10\_sixth.\_3\_19;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: SphereTriangleQuadSurfacenew  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 10:48  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Point3f;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class SphereTriangleQuadSurfacenew extends Applet {  
 public SphereTriangleQuadSurfacenew() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String argv[]) {  
 new MainFrame(new SphereTriangleQuadSurfacenew(), 400, 400);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 1.f, 1.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 transformgroup.addChild(new SphereTriangleArrayDisplay());  
 transformgroup.addChild(new SphereQuadArrayDisplay());  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class SphereTriangleArrayDisplay extends Shape3D {  
 SphereTriangleArrayDisplay() {//定义两个TriangleArray数组  
 float theta1, theta2;//等分角  
 float R = 0.8f;//球体半径  
 int i, j, k;  
 int nn1 = 20, nn2 = 50;//对球体表面的纵向与横向等分点数  
 TriangleArray Trianglesurface1 = new TriangleArray(nn2 \* 3,  
 TriangleArray.COORDINATES | TriangleArray.NORMALS);  
 TriangleArray Trianglesurface2 = new TriangleArray(nn2 \* 3,  
 TriangleArray.COORDINATES | TriangleArray.NORMALS);  
 //定义存放球体体表数据点的数组  
 float[][][] spherexyz = new float[100][200][3];  
 theta1 = (float) Math.PI / nn1;//纵向分角  
 theta2 = 2.0f \* (float) Math.PI / nn2;//横向分角  
 //球体表坐标点计算  
 for (i = 0; i < nn1 + 1; i++)  
 for (j = 0; j < nn2 + 1; j++) {  
 spherexyz[i][j][0] = R \* (float) Math.sin(i \* theta1) \* (float) Math.cos(j \* theta2);  
 spherexyz[i][j][1] = R \* (float) Math.cos(i \* theta1);  
 spherexyz[i][j][2] = R \* (float) Math.sin(i \* theta1) \* (float) Math.sin(j \* theta2);  
 }  
 int c = 0;//以顶点数累加的方式设置点的序号  
 Point3f A01 = new  
 Point3f(spherexyz[0][0][0], spherexyz[0][0][1], spherexyz[0][0][2]);  
 for (j = 0; j < nn2; j++) {  
 Point3f A1 = new Point3f(spherexyz[1][j][0],  
 spherexyz[1][j][1], spherexyz[1][j][2]);  
 Point3f A2 = new Point3f(spherexyz[1][j + 1][0],  
 spherexyz[1][j + 1][1], spherexyz[1][j + 1][2]);  
 Vector3f a = new Vector3f(A1.x - A01.x, A1.y - A01.y, A1.z - A01.z);  
 Vector3f b = new Vector3f(A2.x - A01.x, A2.y - A01.y, A2.z - A01.z);  
 Vector3f n = new Vector3f();  
 n.cross(a, b);  
 n.normalize();  
 //设置点序号及坐标  
 Trianglesurface1.setCoordinate(c, A01);  
 Trianglesurface1.setCoordinate(c + 1, A1);  
 Trianglesurface1.setCoordinate(c + 2, A2);  
 //设置点法向量  
 Trianglesurface1.setNormal(c, n);  
 Trianglesurface1.setNormal(c + 1, n);  
 Trianglesurface1.setNormal(c + 2, n);  
 c = c + 3;  
 }  
 c = 0;  
 Point3f A02 = new Point3f(spherexyz[nn1][0][0],  
 spherexyz[nn1][0][1], spherexyz[nn1][0][2]);  
 for (j = 0; j < nn2; j++) {  
 Point3f A1 = new Point3f(spherexyz[nn1 - 1][j][0],  
 spherexyz[nn1 - 1][j][1], spherexyz[nn1 - 1][j][2]);  
 Point3f A2 = new Point3f(spherexyz[nn1 - 1][j + 1][0],  
 spherexyz[nn1 - 1][j + 1][1], spherexyz[nn1 - 1][j + 1][2]);  
 Vector3f a = new Vector3f(A1.x - A02.x, A1.y - A02.y, A1.z - A02.z);  
 Vector3f b = new Vector3f(A2.x - A02.x, A2.y - A02.y, A2.z - A02.z);  
 Vector3f n = new Vector3f();  
 n.cross(a, b);  
 n.normalize();  
 //设置点序号及坐标  
 Trianglesurface2.setCoordinate(c, A02);  
 Trianglesurface2.setCoordinate(c + 1, A1);  
 Trianglesurface2.setCoordinate(c + 2, A2);  
 //设置点法向量  
 Trianglesurface2.setNormal(c, n);  
 Trianglesurface2.setNormal(c + 1, n);  
 Trianglesurface2.setNormal(c + 2, n);  
 c = c + 3;  
 }  
 this.addGeometry(Trianglesurface1);  
 this.addGeometry(Trianglesurface2);  
 this.setAppearance(createAppearance0());  
 }  
  
 Appearance createAppearance0() {//指定外观  
 PolygonAttributes polygona = new PolygonAttributes();  
 polygona.setCullFace(PolygonAttributes.CULL\_NONE);  
 polygona.setBackFaceNormalFlip(true);  
 //polygona.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 Appearance appearance = new Appearance();  
 appearance.setPolygonAttributes(polygona);  
 Material material = new Material();  
 Color3f red = new Color3f(.0f, 1.0f, 0.0f);  
 material.setDiffuseColor(red);  
 appearance.setMaterial(material);  
 return appearance;  
 }  
}  
  
class SphereQuadArrayDisplay extends Shape3D {  
 SphereQuadArrayDisplay() {//计算球面上点的x、y、z坐标  
 int nn1 = 20, nn2 = 50;  
 float theta1, theta2;  
 float R = 0.8f;  
 int i, j, k;  
 float[][][] spherexyz = new float[100][200][3];  
 theta1 = (float) Math.PI / nn1;  
 theta2 = 2.0f \* (float) Math.PI / nn2;  
 for (i = 0; i < nn1 + 1; i++)  
 for (j = 0; j < nn2 + 1; j++) {  
 spherexyz[i][j][0] = R \* (float) Math.sin(i \* theta1) \* (float) Math.cos(j \* theta2);  
 spherexyz[i][j][1] = R \* (float) Math.cos(i \* theta1);  
 spherexyz[i][j][2] = R \* (float) Math.sin(i \* theta1) \* (float) Math.sin(j \* theta2);  
 }  
 QuadArray Quadsurface = new QuadArray((nn1 - 2) \* nn2 \* 4,  
 GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 int c = 0;//以顶点数累加的方式设置数组中顶点的序号  
 for (i = 1; i < nn1 - 1; i++) {  
 for (j = 0; j < nn2; j++) {//设置一个四边形上的4个点坐标值  
 Point3f A = new Point3f(spherexyz[i][j][0], spherexyz[i][j][1], spherexyz[i][j][2]);  
 Point3f B = new Point3f(spherexyz[i][j + 1][0],  
 spherexyz[i][j + 1][1], spherexyz[i][j + 1][2]);  
 Point3f C = new Point3f(spherexyz[i + 1][j + 1][0],  
 spherexyz[i + 1][j + 1][1], spherexyz[i + 1][j + 1][2]);  
 Point3f D = new Point3f(spherexyz[i + 1][j][0],  
 spherexyz[i + 1][j][1], spherexyz[i + 1][j][2]);  
 //计算四边形的法向量  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
 //设置点的序号  
 Quadsurface.setCoordinate(c, A);  
 Quadsurface.setCoordinate(c + 1, B);  
 Quadsurface.setCoordinate(c + 2, C);  
 Quadsurface.setCoordinate(c + 3, D);  
 //按序号设置点法向量  
 Quadsurface.setNormal(c, n);  
 Quadsurface.setNormal(c + 1, n);  
 Quadsurface.setNormal(c + 2, n);  
 Quadsurface.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 this.addGeometry(Quadsurface);  
 this.setAppearance(createAppearance0());  
 }  
 }  
  
 Appearance createAppearance0() {//指定外观  
 PolygonAttributes polygona = new PolygonAttributes();  
 polygona.setCullFace(PolygonAttributes.CULL\_NONE);  
 polygona.setBackFaceNormalFlip(true);  
 //polygona.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 Appearance appearance = new Appearance();  
 appearance.setPolygonAttributes(polygona);  
 Material material = new Material();  
 Color3f red = new Color3f(1.0f, .0f, .0f);  
 //material.setAmbientColor(red);  
 //material.setSpecularColor(red);  
 material.setDiffuseColor(red);  
 //material.setEmissiveColor(red);  
 //material.setShininess(50.f);  
 appearance.setMaterial(material);  
 return appearance;  
 }  
}

（2）结果截图（球面模型）

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

由三角扇网格、四边网格组成的球面模型

## 实验内容三：

（1）代码

package week10\_sixth.\_4\_1;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: BezierThreeOrderCurve  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 10:50  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Point3f;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class BezierThreeOrderCurve extends Applet {  
 public BezierThreeOrderCurve() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new BezierThreeOrderCurve(), 300, 300);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 Shape3D Bezier = new BezierThreeOrderCurve0();  
 transformgroup.addChild(Bezier);  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class BezierThreeOrderCurve0 extends Shape3D {  
 public BezierThreeOrderCurve0() {  
 int i, j, k;  
 float[] u = new float[100];  
 //下面定义存放[1 u u2 u3]参数矩阵的数组  
 float[][] UU = new float[1][4];  
 //下面定义存放Bezier曲线上点的坐标的数组  
 float[][] curvepoints = new float[100][4];  
 //存放控制顶点坐标的数组  
 float[][] P = {{-0.9f, -0.6f, 0.f, 1.f},  
 {-0.2f, 0.7f, 0.f, 1.f},  
 {0.2f, 0.9f, 0.f, 1.f},  
 {0.9f, -0.7f, 0.f, 1.f}};  
 //存放系数矩阵M的数组  
 float[][] M = {{1.f, 0.f, 0.f, 0.f},  
 {-3.f, 3.f, 0.f, 0.f},  
 {3.f, -6.f, 3.f, 0.f},  
 {-1.f, 3.f, -3.f, 1.f}};  
 int n;//对参数u在[0，1]区间的等分点数  
 float division;//参数u在[0，1]区间的等分线段长度  
 n = 50;  
 division = 1.f / n;  
 for (i = 0; i < n + 1; i++) {  
 u[i] = i \* division;  
 }  
 for (i = 0; i < n + 1; i++) {  
 UU[0][0] = 1.f;  
 UU[0][1] = u[i];  
 UU[0][2] = u[i] \* u[i];  
 UU[0][3] = u[i] \* u[i] \* u[i];  
 matrixm g0 = new matrixm(1, 4, 4, UU, M);  
 matrixm g1 = new matrixm(1, 4, 4, g0.CC, P);  
 curvepoints[i][0] = g1.CC[0][0];  
 curvepoints[i][1] = g1.CC[0][1];  
 curvepoints[i][2] = g1.CC[0][2];  
 curvepoints[i][3] = g1.CC[0][3];  
 //在齐次坐标表示中，前三维的的坐标要除第四维的坐标，才能得到三维坐标系下的坐标值。  
 //在该程序中，第四维的值为1,也可以不除  
 curvepoints[i][0] = g1.CC[0][0] / curvepoints[i][3];  
 curvepoints[i][1] = g1.CC[0][1] / curvepoints[i][3];  
 curvepoints[i][2] = g1.CC[0][2] / curvepoints[i][3];  
 }  
 //定义曲线上点的数组  
 Point3f[] curvepoints0 = new Point3f[100];  
 for (i = 0; i < 100; i++) curvepoints0[i] = new Point3f();  
 for (k = 0; k < n + 1; k++) { //将曲线上点的二维数组转换为Point3f类型的一维数组  
 curvepoints0[k].x = curvepoints[k][0];  
 curvepoints0[k].y = curvepoints[k][1];  
 curvepoints0[k].z = curvepoints[k][2];  
 }  
 //将控制顶点的二维数组转换为Point3f类型的一维数组  
 Point3f[] contralpoints = new Point3f[4];  
 for (i = 0; i < 4; i++) contralpoints[i] = new Point3f();  
 for (i = 0; i < 4; i++) {  
 contralpoints[i].x = P[i][0];  
 contralpoints[i].y = P[i][1];  
 contralpoints[i].z = P[i][2];  
 }  
 int[] contralpointscount = new int[1];  
 int[] curvepointscount = new int[1];  
 contralpointscount[0] = 4;  
 curvepointscount[0] = n + 1;  
 LineStripArray contralpointslines = new  
 LineStripArray(4, LineArray.COORDINATES, contralpointscount);  
 contralpointslines.setCoordinates(0, contralpoints);  
 LineStripArray curvepointslines = new  
 LineStripArray(100, LineArray.COORDINATES, curvepointscount);  
 curvepointslines.setCoordinates(0, curvepoints0);  
 //设置线的属性  
 LineAttributes lineattributes = new LineAttributes();  
 lineattributes.setLineWidth(4.0f);  
 lineattributes.setLineAntialiasingEnable(true);  
 lineattributes.setLinePattern(0);  
 Appearance app = new Appearance();  
 app.setLineAttributes(lineattributes);  
 //针对整个曲线定义颜色，而不是针对顶点定义颜色  
 ColoringAttributes color = new ColoringAttributes();  
 color.setColor(1.f, 0.f, 0.f);  
 app.setColoringAttributes(color);  
 this.addGeometry(curvepointslines);  
 this.addGeometry(contralpointslines);  
 this.setAppearance(app);  
 }  
}  
  
//下面是实现两矩阵相乘的类  
class matrixm {  
 public float CC[][] = new float[4][4];  
 int ll, mm, kk;  
  
 public matrixm(int mmm, int kkk, int nnn, float a[][], float b[][]) {  
 for (ll = 0; ll < mmm; ll++) {  
 for (mm = 0; mm < nnn; mm++) {  
 CC[ll][mm] = 0.f;  
 }  
 }  
 for (ll = 0; ll < mmm; ll++) {  
 for (mm = 0; mm < nnn; mm++) {  
 for (kk = 0; kk < kkk; kk++)  
 CC[ll][mm] = CC[ll][mm] + a[ll][kk] \* b[kk][mm];  
 }  
 }  
 }  
}

（2）结果截图

|  |  |
| --- | --- |
|  |  |
| 图4.1三次Bezier曲线及控制多边形 | 图4.2对u∈[0，1]区间5等分的三次Bezier曲线 |

## 实验内容四：

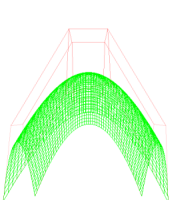
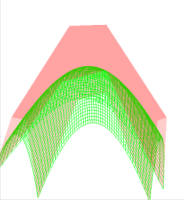
1. 代码

package week10\_sixth.\_4\_2;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: BezierThreeOrderSurface  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-02 10:52  
 \*/  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Point3f;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
public class BezierThreeOrderSurface extends Applet {  
 public BezierThreeOrderSurface() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new BezierThreeOrderSurface(), 400, 400);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 TransformGroup transformgroup = new TransformGroup();  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
 //定义Bezier曲面16个控制顶点的坐标  
 float[][][] P1 = {{{-0.8f, -1.2f, -0.8f, 1.f},  
 {-0.2f, 0.2f, -0.5f, 1.f},  
 {0.2f, 0.3f, -0.5f, 1.f},  
 {0.8f, -1.2f, -0.8f, 1.f}},  
 {{-0.8f, -0.1f, -0.2f, 1.f},  
 {-0.2f, 0.9f, -0.2f, 1.f},  
 {0.2f, 0.9f, -0.2f, 1.f},  
 {0.8f, -0.1f, -0.2f, 1.f}},  
 {{-0.8f, -0.1f, 0.2f, 1.f},  
 {-0.2f, 0.9f, 0.2f, 1.f},  
 {0.2f, 0.9f, 0.2f, 1.f},  
 {0.8f, -0.1f, 0.2f, 1.f}},  
 {{-0.6f, -0.6f, 0.9f, 1.f},  
 {-0.2f, 0.2f, 0.5f, 1.f},  
 {0.2f, 0.3f, 0.5f, 1.f},  
 {0.6f, -0.65f, 0.8f, 1.f}}};  
 //指定Bezier曲面外观属性  
 Appearance app1 = new Appearance();  
 PolygonAttributes polygona1 = new PolygonAttributes();  
 polygona1.setCullFace(PolygonAttributes.CULL\_NONE);  
 //polygona1.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 app1.setPolygonAttributes(polygona1);  
 ColoringAttributes color1 = new ColoringAttributes();  
 color1.setColor(0.f, 1.f, 0.f);  
 app1.setColoringAttributes(color1);  
 //指定Bezier曲面控制顶点网格的外观属性  
 Appearance app2 = new Appearance();  
 PolygonAttributes polygona2 = new PolygonAttributes();  
 polygona2.setCullFace(PolygonAttributes.CULL\_NONE);  
 //polygona2.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 app2.setPolygonAttributes(polygona2);  
 ColoringAttributes color2 = new ColoringAttributes();  
 color2.setColor(2.f, 0.f, 0.f);  
 app2.setColoringAttributes(color2);  
 TransparencyAttributes transparence = new TransparencyAttributes(1, .8f);  
 app2.setTransparencyAttributes(transparence);  
 Shape3D BezierSurfaceface1 = new BezierThreeOrderSurfaceface(P1, app1);  
 transformgroup.addChild(BezierSurfaceface1);  
 Shape3D BezierControlPoints1 = new BezierSurfaceControlPoints(P1, app2);  
 transformgroup.addChild(BezierControlPoints1);  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
//计算Bezier曲面上的点，并生成Bezier曲面  
class BezierThreeOrderSurfaceface extends Shape3D {  
 public BezierThreeOrderSurfaceface(float[][][] P, Appearance app) {  
 int i, j, k;  
 int n0;//定义参数u、v在[0，1]区间的等分点数  
 float division;//定义参数u、v在[0，1]区间的等分线段长度  
 n0 = 50;  
 division = 1.f / n0;  
 //分别定义存放控制顶点x、y、z坐标与第四维的数组  
 float[][] PX = new float[4][4];  
 float[][] PY = new float[4][4];  
 float[][] PZ = new float[4][4];  
 float[][] P4 = new float[4][4];  
 //定义系数矩阵及其转置矩阵  
 float[][] M1 = {{1.f, 0.f, 0.f, 0.f},  
 {-3.f, 3.f, 0.f, 0.f},  
 {3.f, -6.f, 3.f, 0.f},  
 {-1.f, 3.f, -3.f, 1.f}};  
 float[][] M2 = {{1.f, -3.f, 3.f, -1.f},  
 {0.f, 3.f, -6.f, 3.f},  
 {0.f, 0.f, 3.f, -3.f},  
 {0.f, 0.f, 0.f, 1.f}};  
 //定义存放Bezier曲面u、v参数分割点的坐标数组  
 float[][][] UV = new float[n0 + 1][n0 + 1][2];  
 //定义U、V矩阵数组  
 float[][] UU = new float[1][4];  
 float[][] VV = new float[4][1];  
 //定义存放曲面上点的坐标的数组  
 float[][][] SurfaceXYZ = new float[n0 + 1][n0 + 1][4];  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UV[i][j][0] = i \* division;  
 UV[i][j][1] = j \* division;  
 }  
 for (i = 0; i < 4; i++)  
 for (j = 0; j < 4; j++) {  
 PX[i][j] = P[i][j][0];  
 PY[i][j] = P[i][j][1];  
 PZ[i][j] = P[i][j][2];  
 P4[i][j] = P[i][j][3];  
 }  
 //计算曲面上所有点的坐标  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UU[0][0] = 1.f;  
 UU[0][1] = UV[i][j][0];  
 UU[0][2] = UV[i][j][0] \* UV[i][j][0];  
 UU[0][3] = UV[i][j][0] \* UV[i][j][0] \* UV[i][j][0];  
 VV[0][0] = 1.f;  
 VV[1][0] = UV[i][j][1];  
 VV[2][0] = UV[i][j][1] \* UV[i][j][1];  
 VV[3][0] = UV[i][j][1] \* UV[i][j][1] \* UV[i][j][1];  
 //计算一点的x坐标  
 matrixm g0 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g1 = new matrixm(1, 4, 4, g0.CC, PX);  
 matrixm g2 = new matrixm(1, 4, 4, g1.CC, M2);  
 matrixm g3 = new matrixm(1, 4, 1, g2.CC, VV);  
 SurfaceXYZ[i][j][0] = g3.CC[0][0];  
 //计算一点的y坐标  
 matrixm g4 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g5 = new matrixm(1, 4, 4, g4.CC, PY);  
 matrixm g6 = new matrixm(1, 4, 4, g5.CC, M2);  
 matrixm g7 = new matrixm(1, 4, 1, g6.CC, VV);  
 SurfaceXYZ[i][j][1] = g7.CC[0][0];  
 //计算一点的z坐标  
 matrixm g8 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g9 = new matrixm(1, 4, 4, g8.CC, PZ);  
 matrixm g10 = new matrixm(1, 4, 4, g9.CC, M2);  
 matrixm g11 = new matrixm(1, 4, 1, g10.CC, VV);  
 SurfaceXYZ[i][j][2] = g11.CC[0][0];  
 //计算一点的第4维坐标，在该程序中，第4维坐标全为1，可不计算  
 matrixm g12 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g13 = new matrixm(1, 4, 4, g12.CC, P4);  
 matrixm g14 = new matrixm(1, 4, 4, g13.CC, M2);  
 matrixm g15 = new matrixm(1, 4, 1, g14.CC, VV);  
 SurfaceXYZ[i][j][3] = g15.CC[0][0];  
 //将齐次坐标转换为三维坐标系坐标，如果第四维为1，可不除该项  
 SurfaceXYZ[i][j][0] = SurfaceXYZ[i][j][0] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][1] = SurfaceXYZ[i][j][1] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][2] = SurfaceXYZ[i][j][2] / SurfaceXYZ[i][j][3];  
 }  
 QuadArray BeziersurfacecontrolPointsNet = new QuadArray(n0 \* n0 \* 4,  
 GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 int c = 0;//以顶点数累加的方式设置数组中顶点的序号  
 for (i = 0; i < n0; i++) {  
 for (j = 0; j < n0; j++) {//设置一个平面上的4个点  
 Point3f A = new Point3f(SurfaceXYZ[i][j][0], SurfaceXYZ[i][j][1],  
 SurfaceXYZ[i][j][2]);  
 Point3f B = new Point3f(SurfaceXYZ[i][j + 1][0], SurfaceXYZ[i][j + 1][1],  
 SurfaceXYZ[i][j + 1][2]);  
 Point3f C = new Point3f(SurfaceXYZ[i + 1][j + 1][0], SurfaceXYZ[i + 1][j + 1][1],  
 SurfaceXYZ[i + 1][j + 1][2]);  
 Point3f D = new Point3f(SurfaceXYZ[i + 1][j][0], SurfaceXYZ[i + 1][j][1],  
 SurfaceXYZ[i + 1][j][2]);  
 //计算四个点的法向量  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
 //设置点的序号  
 BeziersurfacecontrolPointsNet.setCoordinate(c, A);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 1, B);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 2, C);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 3, D);  
 //按序号设置点的法向量  
 BeziersurfacecontrolPointsNet.setNormal(c, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 1, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 2, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 }  
 this.addGeometry(BeziersurfacecontrolPointsNet);  
 this.setAppearance(app);  
 }  
}  
  
//生成控制顶点网格  
class BezierSurfaceControlPoints extends Shape3D {  
 public BezierSurfaceControlPoints(float[][][] P, Appearance app) {  
 int i, j, k;  
 QuadArray BeziersurfacecontrolPointsNet = new QuadArray(3 \* 3 \* 4, GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 int c = 0;  
 for (i = 0; i < 3; i++) {  
 for (j = 0; j < 3; j++) {  
 Point3f A = new Point3f(P[i][j][0], P[i][j][1], P[i][j][2]);  
 Point3f B = new Point3f(P[i][j + 1][0], P[i][j + 1][1], P[i][j + 1][2]);  
 Point3f C = new Point3f(P[i + 1][j + 1][0], P[i + 1][j + 1][1], P[i + 1][j + 1][2]);  
 Point3f D = new Point3f(P[i + 1][j][0], P[i + 1][j][1], P[i + 1][j][2]);  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
 BeziersurfacecontrolPointsNet.setCoordinate(c, A);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 1, B);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 2, C);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 3, D);  
 BeziersurfacecontrolPointsNet.setNormal(c, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 1, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 2, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 }  
 this.addGeometry(BeziersurfacecontrolPointsNet);  
 this.setAppearance(app);  
 }  
}  
  
class matrixm {  
 public float CC[][] = new float[4][4];  
 int ll, mm, kk;  
  
 public matrixm(int mmm, int kkk, int nnn, float a[][], float b[][]) {  
 for (ll = 0; ll < mmm; ll++){  
 for (mm = 0; mm < nnn; mm++) {  
 CC[ll][mm] = 0.f;  
 }  
 }  
 for (ll = 0; ll < mmm; ll++){  
 for (mm = 0; mm < nnn; mm++) {  
 for (kk = 0; kk < kkk; kk++)  
 CC[ll][mm] = CC[ll][mm] + a[ll][kk] \* b[kk][mm];  
 }  
 }  
 }  
}

（2）结果截图

|  |
| --- |
|  |

修改代码查看线模型：

**三、实验总结**

**实验一总结：**

1.创建一个简单的三维场景，其中包括一个旋转的表面，这个表面由多个四边形组成。

2.设置了背景颜色，以及一个方向光源用于照明。

3.实现了鼠标交互功能，包括旋转、平移和缩放，以便用户可以通过鼠标与三维对象互动。

4.创建了一个自定义的'SurfaceDisplay'类，用于定义表面的几何形状和外观，包括设置多边

形属性和材质颜色。

**实验二总结：**

在本次实验中，实践了一个旋转体的计算和显示程序和实践了一个球面三角网格与四边网格程序显示实例，进一步加深了旋转体设计的原理的理解，自己同时通过局部代码的修改，来查看两个实例的各种模型，逐步了解模型建立的步骤。当然也遇到一些小问题，比如代码打错，容易忘记设置高亮，但在后续都能通过自己的检查修改成功。

1.创建一个简单的三维场景，其中包括一个球体的表面，该表面由不同类型的几何数组表示。

2.设置了背景颜色，以及一个方向光源用于照明。

3.实现了鼠标交互功能，包括旋转、平移和缩放，以便用户可以通过鼠标与三维对象互动。

4.创建了两个自定义的'SphereTriangleArrayDisplay'和`SphereQuadArrayDisplay'类，分别表示球体表面的三角面和四边形面，并设置了外观属性，如颜色和光照。

**实验三总结：**

1. `BezierThreeOrderCurve` 类继承自 `Applet` 类，用于创建和展示三阶贝塞尔曲线的三维可视化。它包括了一个简单的三维场景，其中包含了贝塞尔曲线的可视化对象。

2. `createBranchGroupSceneGraph` 方法创建了场景图（`BranchGroup`），包括了背景、光照、鼠标交互等元素。它还创建了贝塞尔曲线的可视化对象，并将其添加到场景图中。

3. `BezierThreeOrderCurve0` 类继承自 `Shape3D` 类，表示了三阶贝塞尔曲线。它通过计算贝塞尔曲线上的点的坐标，以及绘制控制顶点的线条来创建曲线的可视化。这个类包括以下关键步骤：

- 计算参数 `u` 在 `[0, 1]` 区间的等分点数和线段长度。

- 使用贝塞尔曲线的控制点坐标，计算贝塞尔曲线上每个参数 `u` 对应的坐标。

- 将计算得到的曲线上的点的坐标以及控制点的坐标，转换为 `Point3f` 类型的数组，并创建线条数组表示控制点和曲线。

- 设置线条的属性，包括线宽、抗锯齿和颜色。

- 最后，将这些几何对象添加到 `BezierThreeOrderCurve0` 的外观中。

4. `matrixm` 类是一个用于两矩阵相乘的辅助类，用于计算矩阵相乘的结果。

总体来说，这段代码实现了一个三阶贝塞尔曲线的可视化，并包括了控制点和曲线的绘制。通过 Java 3D 库，你可以在三维空间中创建和呈现这样的曲线，了解如何实现鼠标交互，以及如何使用外观属性来设置线条的样式和颜色。这是一个用于学习三维图形可视化的基本示例。

**实验四总结：**

这段Java代码实现了一个三阶贝塞尔曲面（Bezier Three-Order Surface）的可视化。以下是代码的主要结构和功能：

1. `BezierThreeOrderSurface` 类继承自 `Applet` 类，用于创建和展示三阶贝塞尔曲面的三维可视化。它包括了一个简单的三维场景，其中包含了贝塞尔曲面的可视化对象。

2. `createBranchGroupSceneGraph` 方法创建了场景图（`BranchGroup`），包括了背景、光照、鼠标交互等元素。它还创建了贝塞尔曲面的可视化对象和控制顶点的网格，并将它们添加到场景图中。

3. `BezierThreeOrderSurfaceface` 类继承自 `Shape3D` 类，表示了三阶贝塞尔曲面。它通过计算曲面上的点的坐标以及点的法向量来创建曲面的可视化。这个类包括以下关键步骤：

- 计算参数 `u` 和 `v` 在 `[0, 1]` 区间的等分点数和线段长度。

- 使用贝塞尔曲面的控制点坐标，计算曲面上每个参数 `u` 和 `v` 对应的坐标。

- 计算曲面上点的法向量，以便渲染阴影和光照效果。

- 创建 `QuadArray` 以表示曲面的网格，并将其中的坐标和法向量设置为曲面上点的值。

- 设置曲面的外观属性，包括颜色和抗锯齿。

4. `BezierSurfaceControlPoints` 类用于创建控制顶点的网格，类似于 `BezierThreeOrderSurfaceface` 类，但它只绘制控制顶点而不计算法向量。

5. `matrixm` 类是一个用于两矩阵相乘的辅助类，用于计算矩阵相乘的结果。

总体来说，这段代码实现了一个三阶贝塞尔曲面的可视化，包括控制顶点和曲面网格的绘制。这是一个用于学习三维图形可视化的高级示例，涵盖了曲面计算、鼠标交互、光照、法向量计算等方面。通过 Java 3D 库，你可以在三维空间中创建和呈现这样的曲面，用于学习计算机图形学的基本概念和技术。

**《计算机图形学》实验报告七**

**学生姓名：李季鸿 班级：21级计科3班 学号：20213002624**

**实验地点：9-202 指导教师：高新瑞**

**实验日期：**2023.11.09  **实验课时：2学时**

**实验环境：**Windows 10+JDK1.8+记事本+IntelliJ IDEA

**一、实验目的**

**实验一目的：**

学习如何使用两个双三次Bezier曲面拼接编程。

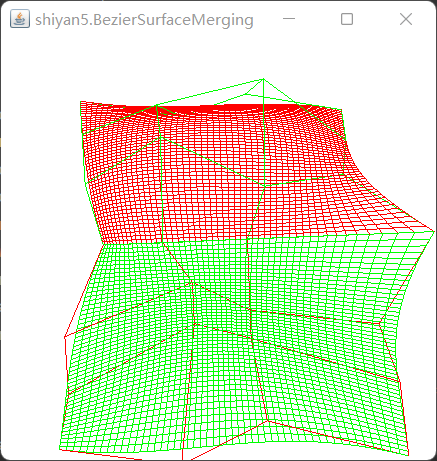
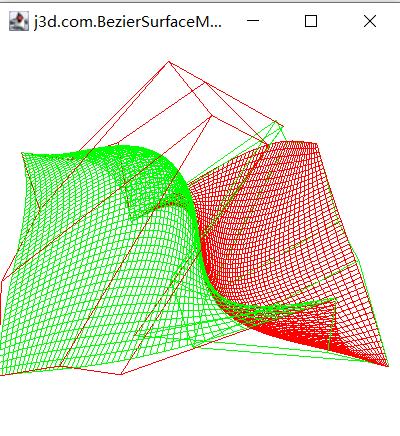
1. **实验过程**

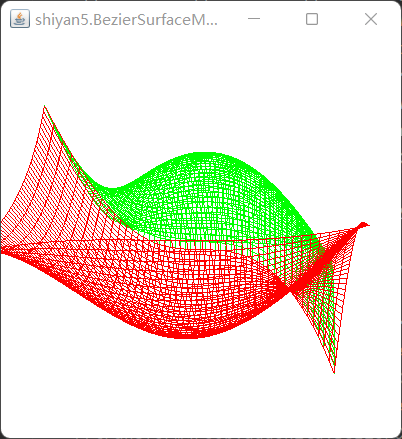
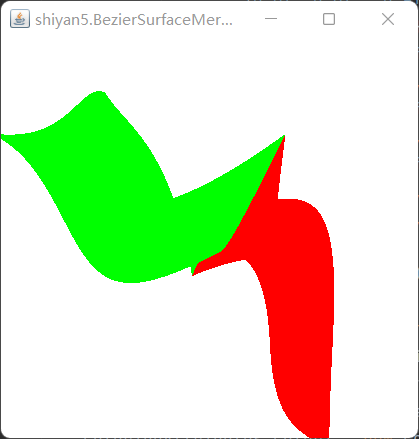
## 实验内容一：

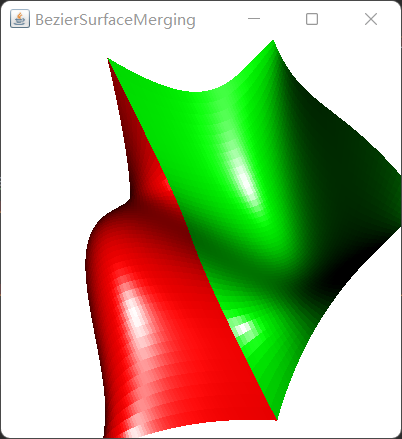
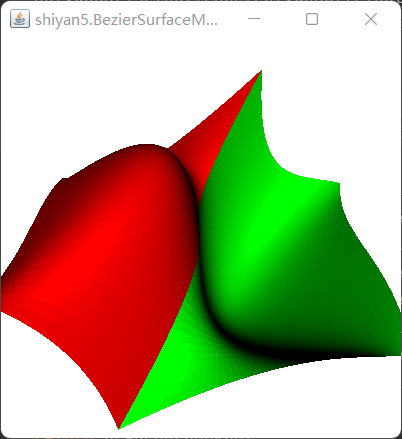
（1）代码

package week11\_seventh.\_4\_3;  
  
import com.sun.j3d.utils.applet.MainFrame;  
import com.sun.j3d.utils.behaviors.mouse.MouseRotate;  
import com.sun.j3d.utils.behaviors.mouse.MouseTranslate;  
import com.sun.j3d.utils.behaviors.mouse.MouseZoom;  
import com.sun.j3d.utils.universe.SimpleUniverse;  
  
import javax.media.j3d.\*;  
import javax.vecmath.Color3f;  
import javax.vecmath.Point3d;  
import javax.vecmath.Point3f;  
import javax.vecmath.Vector3f;  
import java.applet.Applet;  
import java.awt.\*;  
  
/\*\*  
 \* \\* Created with IntelliJ IDEA.  
 \* \\* @ProjectName: Computer graphics  
 \* \\* @FileName: BezierSurfaceMerging  
 \* \\* @author: li-jihong  
 \* \\* Date: 2023-11-09 19:24  
 \*/  
public class BezierSurfaceMerging extends Applet {  
 public BezierSurfaceMerging() {  
 setLayout(new BorderLayout());  
 GraphicsConfiguration gc = SimpleUniverse.getPreferredConfiguration();  
 Canvas3D c = new Canvas3D(gc);  
 add("Center", c);  
 BranchGroup BranchGroupScene = createBranchGroupSceneGraph();  
 SimpleUniverse u = new SimpleUniverse(c);  
 u.getViewingPlatform().setNominalViewingTransform();  
 u.addBranchGraph(BranchGroupScene);  
 }  
  
 public static void main(String[] args) {  
 new MainFrame(new BezierSurfaceMerging(), 400, 400);  
 }  
  
 public BranchGroup createBranchGroupSceneGraph() {  
 BranchGroup BranchGroupRoot = new BranchGroup();  
 BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0), 100.0);  
 Color3f bgColor = new Color3f(1.0f, 1.0f, 1.0f);  
 Background bg = new Background(bgColor);  
 bg.setApplicationBounds(bounds);  
 BranchGroupRoot.addChild(bg);  
 Color3f directionalColor = new Color3f(1.f, 0.f, 0.f);  
 Vector3f vec = new Vector3f(0.f, 0.f, -1.0f);  
 DirectionalLight directionalLight = new DirectionalLight(directionalColor, vec);  
 directionalLight.setInfluencingBounds(bounds);  
 BranchGroupRoot.addChild(directionalLight);  
 Transform3D tr = new Transform3D();  
 tr.setScale(0.85);  
 TransformGroup transformgroup = new TransformGroup(tr);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_WRITE);  
 transformgroup.setCapability(TransformGroup.ALLOW\_TRANSFORM\_READ);  
 BranchGroupRoot.addChild(transformgroup);  
 MouseRotate mouserotate = new MouseRotate();  
 mouserotate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mouserotate);  
 mouserotate.setSchedulingBounds(bounds);  
 MouseZoom mousezoom = new MouseZoom();  
 mousezoom.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousezoom);  
 mousezoom.setSchedulingBounds(bounds);  
 MouseTranslate mousetranslate = new MouseTranslate();  
 mousetranslate.setTransformGroup(transformgroup);  
 BranchGroupRoot.addChild(mousetranslate);  
 mousetranslate.setSchedulingBounds(bounds);  
//定义第一个Bezier曲面的16个控制顶点  
 float[][][] P1 = {{{-0.8f, 0.9f, -0.4f, 1.f},  
 {-0.2f, 0.8f, -0.5f, 1.f},  
 {0.2f, 0.9f, -0.4f, 1.f},  
 {0.8f, 0.8f, -0.5f, 1.f}},  
 {{-0.8f, 0.7f, -0.4f, 1.f},  
 {-0.2f, 0.6f, 0.9f, 1.f},  
 {0.2f, 0.7f, 0.8f, 1.f},  
 {0.8f, 0.6f, -0.4f, 1.f}},  
 {{-0.8f, 0.4f, -0.4f, 1.f},  
 {-0.2f, 0.5f, 0.8f, 1.f},  
 {0.2f, 0.3f, 0.7f, 1.f},  
 {0.8f, 0.4f, -0.5f, 1.f}},  
 {{-0.8f, 0.f, -0.8f, 1.f},  
 {-0.2f, 0.1f, 0.9f, 1.f},  
 {0.2f, 0.f, -0.8f, 1.f},  
 {0.8f, 0.1f, 0.9f, 1.f}}};  
//定义第一个Bezier曲面外观属性  
 Appearance app1 = new Appearance();  
 PolygonAttributes polygona1 = new PolygonAttributes();  
 polygona1.setBackFaceNormalFlip(true);  
 polygona1.setCullFace(PolygonAttributes.CULL\_NONE);  
 polygona1.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 app1.setPolygonAttributes(polygona1);  
 ColoringAttributes color1 = new ColoringAttributes();  
 color1.setColor(1.f, 0.f, 0.f);  
 app1.setColoringAttributes(color1);  
//定义第二个Bezier曲面的16个控制顶点  
 float[][][] P2 = {{{-0.8f, 0.f, -0.8f, 1.f},  
 {-0.2f, 0.1f, 0.9f, 1.f},  
 {0.2f, 0.f, -0.8f, 1.f},  
 {0.8f, 0.1f, 0.9f, 1.f}},  
 {{-0.8f, -0.4f, 0.2f, 1.f},  
 {-0.2f, -0.3f, -0.9f, 1.f},  
 {0.2f, -0.5f, -0.8f, 1.f},  
 {0.8f, -0.4f, 0.2f, 1.f}},  
 {{-0.8f, -0.7f, 0.2f, 1.f},  
 {-0.2f, -0.6f, -0.9f, 1.f},  
 {0.2f, -0.7f, -0.8f, 1.f},  
 {0.8f, -0.6f, 0.5f, 1.f}},  
 {{-0.8f, -0.9f, 0.4f, 1.f},  
 {-0.2f, -0.9f, 0.6f, 1.f},  
 {0.2f, -0.8f, 0.4f, 1.f},  
 {0.8f, -0.9f, 0.6f, 1.f}}};  
//定义第二个Bezier曲面外观属性  
 Appearance app2 = new Appearance();  
 PolygonAttributes polygona2 = new PolygonAttributes();  
 polygona2.setBackFaceNormalFlip(true);  
 polygona2.setCullFace(PolygonAttributes.CULL\_NONE);  
 polygona2.setPolygonMode(PolygonAttributes.POLYGON\_LINE);  
 app2.setPolygonAttributes(polygona2);  
 ColoringAttributes color2 = new ColoringAttributes();  
 color2.setColor(0.f, 1.f, 0.f);  
 app2.setColoringAttributes(color2);  
 Shape3D BezierSurfaceface1 = new BezierThreeOrderSurfaceface(P1, app1);  
 transformgroup.addChild(BezierSurfaceface1);  
 Shape3D BezierSurfaceface2 = new BezierThreeOrderSurfaceface(P2, app2);  
 transformgroup.addChild(BezierSurfaceface2);  
 Shape3D BezierControlPoints1 = new BezierSurfaceControlPoints(P1, app2);  
 transformgroup.addChild(BezierControlPoints1);  
 Shape3D BezierControlPoints2 = new BezierSurfaceControlPoints(P2, app1);  
 transformgroup.addChild(BezierControlPoints2);  
 BranchGroupRoot.compile();  
 return BranchGroupRoot;  
 }  
}  
  
class BezierThreeOrderSurfaceface extends Shape3D {  
 public BezierThreeOrderSurfaceface(float[][][] P, Appearance app) {  
 int i, j, k;  
 int n0;//定义对参数u、v在[0，1]区间的等分点数  
 float division;//参数u在[0，1]区间的等分线段长度  
 n0 = 50;  
 division = 1.f / n0;  
//分别定义存放控制顶点x、y、z坐标与第四维坐标的数组  
 float[][] PX = new float[4][4];  
 float[][] PY = new float[4][4];  
 float[][] PZ = new float[4][4];  
 float[][] P4 = new float[4][4];  
//定义系数矩阵及其转置矩阵  
 float[][] M1 = {{1.f, 0.f, 0.f, 0.f},  
 {-3.f, 3.f, 0.f, 0.f},  
 {3.f, -6.f, 3.f, 0.f},  
 {-1.f, 3.f, -3.f, 1.f}};  
 float[][] M2 = {{1.f, -3.f, 3.f, -1.f},  
 {0.f, 3.f, -6.f, 3.f},  
 {0.f, 0.f, 3.f, -3.f},  
 {0.f, 0.f, 0.f, 1.f}};  
//定义Bezier曲面的u、v参数分割点坐标数组  
 float[][][] UV = new float[n0 + 1][n0 + 1][2];  
//定义U、V矩阵数组  
 float[][] UU = new float[1][4];  
 float[][] VV = new float[4][1];  
//定义存放曲面上点的坐标的数组  
 float[][][] SurfaceXYZ = new float[n0 + 1][n0 + 1][4];  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UV[i][j][0] = i \* division;  
 UV[i][j][1] = j \* division;  
 }  
 for (i = 0; i < 4; i++)  
 for (j = 0; j < 4; j++) {  
 PX[i][j] = P[i][j][0];  
 PY[i][j] = P[i][j][1];  
 PZ[i][j] = P[i][j][2];  
 P4[i][j] = P[i][j][3];  
 }  
//计算曲面上点的坐标值  
 for (i = 0; i < n0 + 1; i++)  
 for (j = 0; j < n0 + 1; j++) {  
 UU[0][0] = 1.f;  
 UU[0][1] = UV[i][j][0];  
 UU[0][2] = UV[i][j][0] \* UV[i][j][0];  
 UU[0][3] = UV[i][j][0] \* UV[i][j][0] \* UV[i][j][0];  
 VV[0][0] = 1.f;  
 VV[1][0] = UV[i][j][1];  
 VV[2][0] = UV[i][j][1] \* UV[i][j][1];  
 VV[3][0] = UV[i][j][1] \* UV[i][j][1] \* UV[i][j][1];  
 //计算一点的x坐标  
 matrixm g0 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g1 = new matrixm(1, 4, 4, g0.CC, PX);  
 matrixm g2 = new matrixm(1, 4, 4, g1.CC, M2);  
 matrixm g3 = new matrixm(1, 4, 1, g2.CC, VV);  
 SurfaceXYZ[i][j][0] = g3.CC[0][0];  
 //计算一点的y坐标  
 matrixm g4 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g5 = new matrixm(1, 4, 4, g4.CC, PY);  
 matrixm g6 = new matrixm(1, 4, 4, g5.CC, M2);  
 matrixm g7 = new matrixm(1, 4, 1, g6.CC, VV);  
 SurfaceXYZ[i][j][1] = g7.CC[0][0];  
 //计算一点的z坐标  
 matrixm g8 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g9 = new matrixm(1, 4, 4, g8.CC, PZ);  
 matrixm g10 = new matrixm(1, 4, 4, g9.CC, M2);  
 matrixm g11 = new matrixm(1, 4, 1, g10.CC, VV);  
 SurfaceXYZ[i][j][2] = g11.CC[0][0];  
 //计算一点的第4维坐标  
 matrixm g12 = new matrixm(1, 4, 4, UU, M1);  
 matrixm g13 = new matrixm(1, 4, 4, g12.CC, P4);  
 matrixm g14 = new matrixm(1, 4, 4, g13.CC, M2);  
 matrixm g15 = new matrixm(1, 4, 1, g14.CC, VV);  
 SurfaceXYZ[i][j][3] = g15.CC[0][0];  
 //将齐次坐标转换为三维坐标系坐标，如果第4维为1，则不用除第4维  
 SurfaceXYZ[i][j][0] = SurfaceXYZ[i][j][0] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][1] = SurfaceXYZ[i][j][1] / SurfaceXYZ[i][j][3];  
 SurfaceXYZ[i][j][2] = SurfaceXYZ[i][j][2] / SurfaceXYZ[i][j][3];  
 }  
 QuadArray BezierQuadsurfaceface = new QuadArray(n0 \* n0 \* 4,  
 GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 int c = 0;//以顶点数累加的方式设置顶点的序号  
 for (i = 0; i < n0; i++) {  
 for (j = 0; j < n0; j++) {//设置一个平面上的4个点  
 Point3f A = new Point3f(SurfaceXYZ[i][j][0], SurfaceXYZ[i][j][1],  
 SurfaceXYZ[i][j][2]);  
 Point3f B = new Point3f(SurfaceXYZ[i][j + 1][0], SurfaceXYZ[i][j + 1][1],  
 SurfaceXYZ[i][j + 1][2]);  
 Point3f C = new Point3f(SurfaceXYZ[i + 1][j + 1][0], SurfaceXYZ[i + 1][j + 1][1],  
 SurfaceXYZ[i + 1][j + 1][2]);  
 Point3f D = new Point3f(SurfaceXYZ[i + 1][j][0], SurfaceXYZ[i + 1][j][1],  
 SurfaceXYZ[i + 1][j][2]);  
//计算由四个点形成的平面的法向量  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
//设置点的序号  
 BezierQuadsurfaceface.setCoordinate(c, A);  
 BezierQuadsurfaceface.setCoordinate(c + 1, B);  
 BezierQuadsurfaceface.setCoordinate(c + 2, C);  
 BezierQuadsurfaceface.setCoordinate(c + 3, D);  
//设置点的法向量  
 BezierQuadsurfaceface.setNormal(c, n);  
 BezierQuadsurfaceface.setNormal(c + 1, n);  
 BezierQuadsurfaceface.setNormal(c + 2, n);  
 BezierQuadsurfaceface.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 }  
 this.addGeometry(BezierQuadsurfaceface);  
 this.setAppearance(app);  
 }  
}  
  
class BezierSurfaceControlPoints extends Shape3D {  
 public BezierSurfaceControlPoints(float[][][] P, Appearance app) {  
 int i, j, k;  
 QuadArray BeziersurfacecontrolPointsNet = new QuadArray(3 \* 3 \* 4,  
 GeometryArray.COORDINATES | GeometryArray.NORMALS);  
 int c = 0;  
 for (i = 0; i < 3; i++) {  
 for (j = 0; j < 3; j++) {  
 Point3f A = new Point3f(P[i][j][0], P[i][j][1], P[i][j][2]);  
 Point3f B = new Point3f(P[i][j + 1][0], P[i][j + 1][1], P[i][j + 1][2]);  
 Point3f C = new Point3f(P[i + 1][j + 1][0], P[i + 1][j + 1][1], P[i + 1][j + 1][2]);  
 Point3f D = new Point3f(P[i + 1][j][0], P[i + 1][j][1], P[i + 1][j][2]);  
 Vector3f a = new Vector3f(A.x - B.x, A.y - B.y, A.z - B.z);  
 Vector3f b = new Vector3f(C.x - B.x, C.y - B.y, C.z - B.z);  
 Vector3f n = new Vector3f();  
 n.cross(b, a);  
 n.normalize();  
 BeziersurfacecontrolPointsNet.setCoordinate(c, A);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 1, B);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 2, C);  
 BeziersurfacecontrolPointsNet.setCoordinate(c + 3, D);  
 BeziersurfacecontrolPointsNet.setNormal(c, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 1, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 2, n);  
 BeziersurfacecontrolPointsNet.setNormal(c + 3, n);  
 c = c + 4;  
 }  
 }  
 this.addGeometry(BeziersurfacecontrolPointsNet);  
 this.setAppearance(app);  
 }  
}  
  
class matrixm {  
 public float CC[][] = new float[4][4];  
 int ll, mm, kk;  
  
 public matrixm(int mmm, int kkk, int nnn, float a[][], float b[][]) {  
 for (ll = 0; ll < mmm; ll++)  
 for (mm = 0; mm < nnn; mm++) {  
 CC[ll][mm] = 0.f;  
 }  
 for (ll = 0; ll < mmm; ll++)  
 for (mm = 0; mm < nnn; mm++) {  
 for (kk = 0; kk < kkk; kk++) CC[ll][mm] = CC[ll][mm] + a[ll][kk] \* b[kk][mm];  
 }  
 }  
}

**三、实验总结**

结果截图（四边网格曲面模型）

两张双三次Bezier曲面的拼接及不同光照效果

**三、实验总结**

通过本次实验，进一步巩固理解了双三次Bezier曲面的定义，以及如何将双三次Bezier曲面的矩阵表示应用到java3D编程中，关于理论知识个人将其总结在第二部分。本次实验不仅需要理解和读懂代码，更要自己修改代码。要将Bezier曲线连接成曲面，然后设置高亮颜色，这是最关键的部分。连接成曲面只需要注释掉以下代码：

//polygona1.setPolygonMode(PolygonAttributes.POLYGON\_LINE);

//polygona2.setPolygonMode(PolygonAttributes.POLYGON\_LINE);

其中设置高亮最为关键的代码如下：

Material ma1 = new Material();  
ma1.setSpecularColor(1f, 0f, 0f);  
ma1.setDiffuseColor(1f, 0f, 0f);  
app1.setMaterial(ma1);